ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 4, June 2012

Design of Multi-Channel UART Controller Based On FIFO and FPGA

[1]Deepchand Jaiswal
M.Tech Research scholar
EC Dept. LNCT-Bhopal, India
. deepchand55@gmail.com

[2]Dr. Rita Jain
E.C. Department
LNCT-Bhopal, India

[3]Prof. M. Zahid Alam
E.C. Department
LNCT-Bhopal, India

Abstract: ***This paper presents a multi-channel UART controller based on FPGA (Field Programmable Gate Array). UART a kind of serial communication circuit is used widely. A universal asynchronous receive/transmit (UART) is an integrated circuit which plays the most important role in serial communication. The architecture of the system is introduced. The flow charts of data processing as well as the implementation state machine are also presented in detail. The controller can be used to implement communications in complex system with different Baud Rates of sub-controllers. The controller is reconfigurable and scalable 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.***

Keywords: ***FIFO, FPGA, UART, MULTI-CHANNEL,***

I. INTRODUCTION

UART is a popular methodology of serial asynchronous communication. The UART data frame is composed of a start bit of one bit-length logic 0，5 to 8 bit-length data bits and a stop bit of one. one and a half or two bit length. There may be one bit-length checkout bit after the data bits if it is necessary. And its characteristic is that the frame contains only one character and it is transmitted continuously one by one. Typically, the UART is connected between a micro-processor and a peripheral. Although UART is popular and the structure of the flame is simple, it is inefficient. In this paper, three disadvantages, which influence its efficiency, are pointed out and solved.

(1) Usually, there is only one channel which can only connect to a single peripheral in normal UART controller. In this case the number of chips will increase with the augment of peripherals, which may cost a lot of space and resources

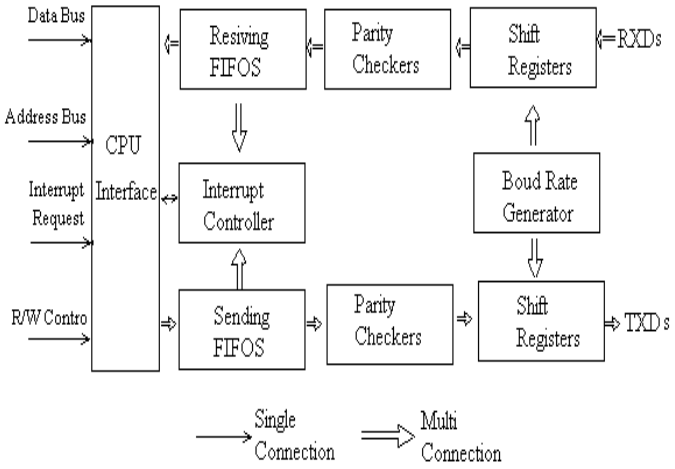

Fig. 1 System architecture

(2) In general the interrupt request is an easy and common method to notify the microprocessor of a flame's operation. But there comes a problem that the burden of the processor will become heavier and heavier when the interrupt request occurs more and more frequently, i.e. when there are frequent interrupt requests and only a few characters are transmitted during each interrupt time, the processor's implementation efficiency will becomeex,, tersely low, It means that the processor

would spend a lot of time on processing small amount of data without doing other necessary tasks.

(3) Although certain processor's data bus is 32．bit wide or more, the traditional UAI 'controller transmits only one byte data to the microprocessor with which it connected at a time. In this case，24 bits or more of data bus are wasted．

To solve the first problem more UART channels in a single chip are required and the same as ST16C554t"does．However, this scheme may result in more frequent interrupts which will make the second problem more serious. Considering both of them, the issues are focused on the improvement of data transmission efficiency during interrupt times． So in this paper, the ideas of enhanced parallel processing of data, novel interrupt controlling mechanism. As well as utilizing the whole data bus width are put forward. Realization is implemented using Verilog HDL on FPGA．

II. SYSTEM ARCHITECTURE AND DESIGN PRINCIPLE DESCRIPTION

As a UART controller, the service affects are micro-processors and serial communication terminations (SCT) . To be simple, it is assumed that the bus of microprocessor is the same as 32．bit Motorola Power PCT series and the SCT only has two transmission wires, TXD (for transmitted data) and RXD (for received data) . Therefore the main system, architecture of the controller is shown in Fig．1 which is based on the assumption presented here an d UART protoco1. The architecture introduced in Fig.1 can be viewed as a description of one channel and the others. In the controller are exactly the same.

2.1 Modules of receiving flow

The modules are shift registers, parity checkers and receiving FIFOs. The shift registers connected to RXD wires are used to transfer all inputting serial data into parallel data. Which will be sent to parity checkers as soon as a correct frame is received and they also check the integrality of the frames? The parity checkers are used to implement parity check, if it is necessary and inform the FIFOs whether the receiving frames are correct. The receiving FIFOs will store the data which are correctly received until micro-processor intends to read them．

2.2 Modules of sending flow

The modules are sending FIFOs parity checkers and shift registers. The sending FIFOs will store data sent from microprocessor while there still are data under transmission on the TXD wires. The parity checkers here are used to attach the start and stop bits to the arriving frames as well as filling the checkout bit into frames if it is necessary. Then the frames will be sent out serially by the shift registers through TXD wires.

2.3 Other components

The baud-rate generator generates different baud-rate clocks used as time benchmarks in UART transmission. The CPU interface manages communications, including buses control interrupts， reading signal, writing signal etc. between the microprocessor and the controller. The interrupt controller takes responsibility of managing the interrupt requests generated by each channel and sending them to microprocessor based on certain principles． To improve the system efficiency, the design is focused on how to accomplish more tasks in less time. In other words more implementations in fewer system clock periods are expected. There are two methods adopted in this design, implementing correlative tasks in parallel as many as possible and reducing the implementation steps

of the ordinal task. The first method is recommended when there are weak constraints of time relationship among tasks so it is adopted in the module design s of shift registers parity checkers and FIFOs. The other method works well when the constraints are strong such as responding continuous interrupts on one wire which should be implemented one by one. Therefore it is adopted in interrupt controller module.

III. SENDING AND RECEIVING FLOWS DESIGN

There are three clocks that should be considered in the design the first one is the system main clock which is the basal time scale of all the tasks and it is very fast, the clock is 50 MHz in this design: the second one is the baud-rate clock which is the division of the main clock, it is only used when there are requirements to send data or to synchronize receiving data, the third one is the clock of microprocessor instruction. Because the baud rate clock and instruction clock are far slower than the main clock, the pre-operations which are driven by the main clock can be implemented during receiving process and sending process, and it is possible to transmit data as soon as it is ready. Considering the second problem mentioned in Section l, it can be concluded that the time of data transmission between controllers and microprocessor will be reduced if all the 32 bits of data bus are utilized。111is also is adopted in the design. Using these methods, all the tasks will be accomplished efficiently.

3.1 Module cooperation in sending flow

Based on the description in Section 2, the data flow of sending process is shown in Fig.2. The conclusion of Fig.2 is that there are two serial operations in the process of frame composing which are the system choke points (the first operation is indicated with label l and the second one is indicated with label 2 in Fig.2. he first one is due to Sending serially operation, so if a notification is feed back to active the Writing into reparation, at the same time that the frame is delivered this chock point will be eliminated.

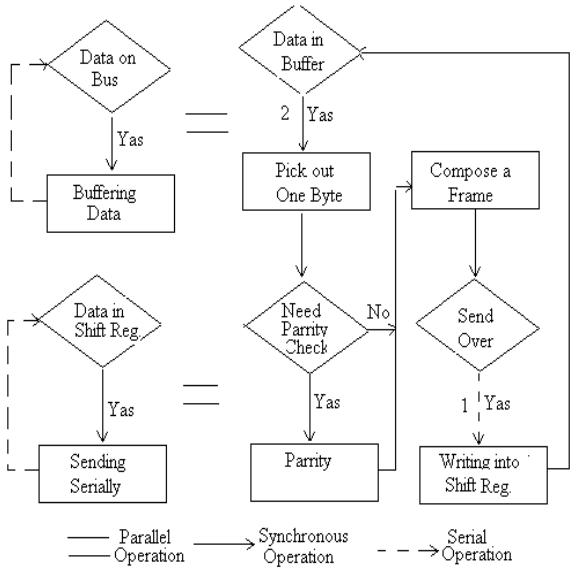

Fig. 2 Operation flow of sending process

The state machine of Sending serially process is shown in Fig. 3. The state machine as well as the counters in the Stop and the Notify states is driven by the baud rate clock

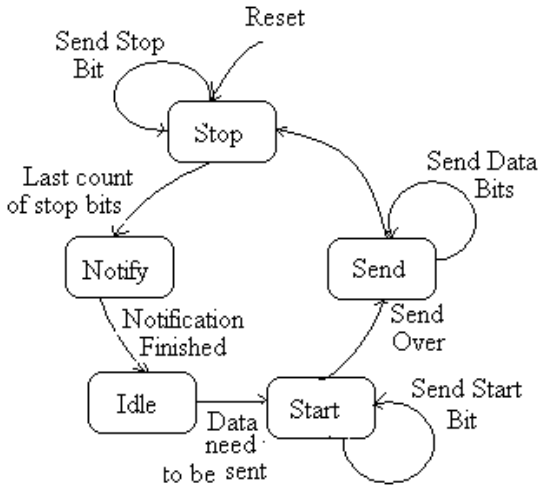

Fig 3 State machine of sending serially

At the last count of the Stop counter, the state machine would switch to the Notify state from the Stop state. The Notify state in which the notification is generated and the last stop bit ends, will last one baud rate clock. Then the state machine will switch to the idle state. Because the main clock, which is the benchmark of the frame composing, is far faster than the baud-rate clock, a new frame will be ready as soon as the Idle state occurs. All of this assures that no time is needed to wait for a new frame's preparation, so the first chock point is eliminate.

3.2 Sending FIFO system design

The instruction clock of microprocessor is also faster than the baud rate clock so the data from micro-processor usually arrive at the moment when there are data under transmission. To prevent system from waiting all though the transmission, which is the second chock point in Fig. 2, the fresh data should be buffered during the transmission. FIFO is adopted in the design . and its width is 32-bit which is referred to the width of data bus. In this paper, RAM integrated in FPGA is used to realize the design of the FIFO to reduce the usage of FPGA logic elements (LEs). Architecture of the sending FIFO system is shown in Fig. 4

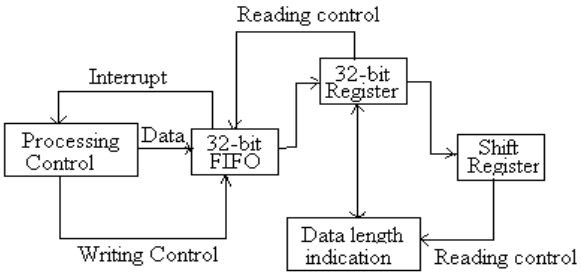

Fig 4 Architecture of sending FIFO system

In order to satisfy the synthesizable condition of Verilog HDL RAM , the input bit width and the output one of the FIFO should be the same. However, the Sh register, whose length is only a frame length, is shorter than 32 bits, so a 32-bit register composed of LEs is required, which will be used as a buffer between the FIFO and the Shift register. It is quite necessary for the Indicator to record how many bytes of data are in the 32-bit register. That is because when there are less than 4 bytes of data in the 32-bit register, such as the last time of transmission, the Shift register should obtain the exact amount of bytes to be transmitted. Thus the implementation flow of the whole 32-bit width FIFO system is shown in Fig. 5

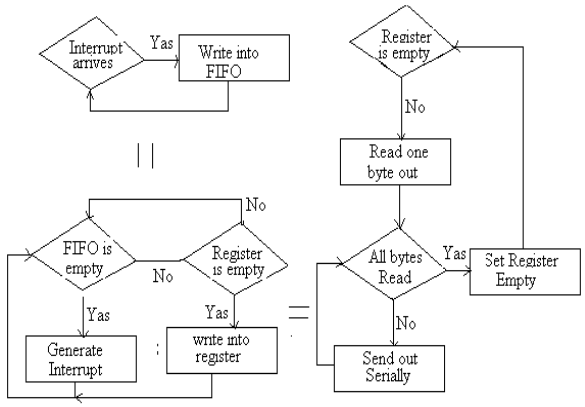

Fig 5 Sending FIFO implementation flow

It is indicated in Fig. 5 that the writing process of micro-processor can implement ceaselessly no mater whether the previous data have been sent or not which means that more times of writing process can be implemented during each interrupt time. And during each time at most four bytes of data can be written now, compared with the input of one byte of data using traditional method. So the time required for a whole task is reduced.

3.3 Module cooperation in receiving flow

There is an important difference between the receiving flows and the sending flow. It is that the transmission

velocity of the data source (microprocessor) is far faster than that of the data destination (serial periphera1) in the sending flow, while the situation is just the opposite in the receiving flow. So the data from microprocessor should be delivered to peripheral immediately to prevent data cumuli in the sending flow. But in the receiving flow, it is possible for the controller to buffer data until microprocessor has time to respond to the receiving request, which can be used to solve the second problem mentioned in Section 1. Thus the requirement of the reduction of data processing time in receiving flow is focused on the Shift register module and parity checker module in Fig.1. The state machine is shown in Fig.6 and it is driven by baud-rate clock which is used to sample input signals.

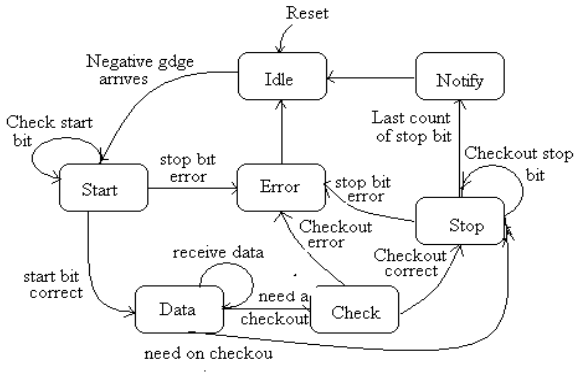

Fig 6 State machine of receiving flow

In the state machine, the Check state is between the Data state and the Stop state and it takes the responsibility of parity checkout. Such design is based on the fact that the baud, rate clock is designed as 16 times faster as input signal so there are enough clock periods to check parity during the arrival of stop bits, therefore, the parity checkout can implement in parallel with the signal sampling process. The state machine will switch to the Notify state at the last count of the stop bits. The Notify state takes the responsibility of notifying FIFOs to store received data. The principle is the same as the Check state an d the clock period would also be reduced In this way All of these designs ensure that the correct data can be stored in FIFOs as soon as a frame is received completely, an d in this way, the system efficiency is improved.

3.4 Receiving FIFO design

Different from the sending FIFO which is used to prevent data cumuli the receiving FIFO focuses on reducing the interrupt frequency. In the design , receiving interrupts are generated only when the quantity of the data stored in FIFO goes beyond a threshold value or it is time out for the FIFO to wait for the microprocessor to receive data. The threshold is a certain value between zeros an d FIFO depth. The architecture of receiving FIFO system is shown in Fig.7

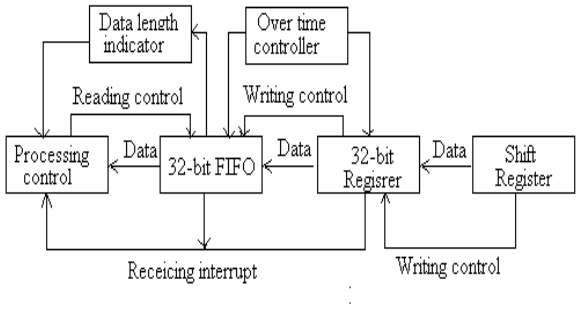

Fig 7 Architecture of receiving FIFO system

In the receiving flow, the 32—bit register and the Data length indicator operate in the same manner as the sending FIFO does, an d the whole 32—bit bus can be utilized to increase the data transmission velocity. The Overtime controller is active only when there still are data in the FIFO or the register, an d its amount are not beyond the threshold. In this case an overtime interrupt will be generated to notify the micro--processor of receiving request if the counter of overtime controller is overflow. The whole implementation flow is shown in Fig. 8. To reduce the reading times of the

microprocessor, the Data length indicator only needs to be accessed during the overtime interrupt.

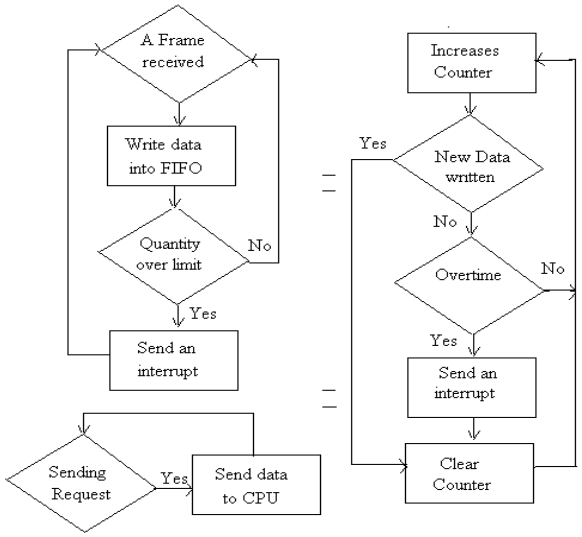

Fig 8 Receiving FIFO implementation Flow

This is because the quantity of data stored in 32—bit FIFO must be the multiple of 4 bytes on the arrival of over-threshold interrupt To conclude, the parallel implementations are optimized an d FIFOs are redesigned to enhance transmission velocity by utilizing the data bus adequately. Therefore, the efficiency is improved an d more channels are able to cooperate in the controller

IV. INTERRUPT CONTROLLER DESIGN

4.1 Scheme argumentation

The interrupt controller cooperates with both internal channels an d external micro—processor, and the efficiency of internal channels is assured by design s in Section 3. So in this section, attentions should be paid to the implementation of microprocessor who processes tasks Interrupt service routine(ISR)is implemented by microprocessor. Besides the reading an d the writing processes, the most necessary task of ISR is to obtain interrupt source an d interrupt state. So if the steps of these processes are reduced, the efficiency of the ISR will be improved In the realization of the controller, 16 UART channels are included, which means that there are 16 coequal interrupt sources to the interrupt controller. Because of the coequality, the principle of interrupt management is first in first serves (FIFS), and interrupt nesting is forbidden. So there comes a problem about how to deal with the interrupts when more than one channel send interrupt requests to the controller at the same time. The first method is to arrange these interrupts according to certain principle (it will be indicated later in this section) and then send them out in order. The disadvantage is that there are several interrupts generated so that the processor has to switch into ISR frequently. Then the processor should poll over to implement relevant operations. It will lead to a mass of time overhead in polling operations.

4.2 Scheme design

Based on the principles mentioned above, the architecture of the interrupt controller is shown in Fig.9 which contains a 256-bit deep and bit wide FIFO. In the design of all the channels the shortest time between two ordinal interrupt requests is only one system clock period. So if severa1 interrupts occur at the same time, they should be written into the FIFO in one clock period. To satisfy this condition,

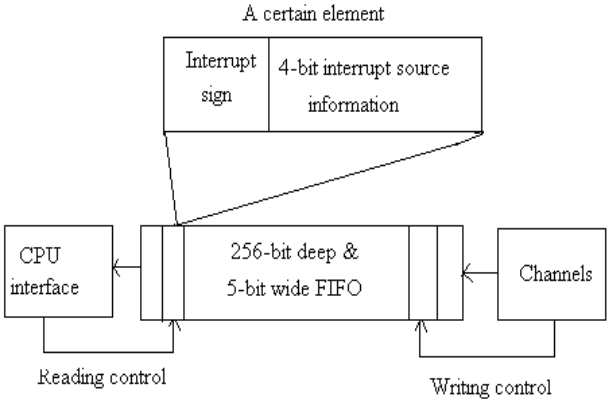

Fig. 9 Interrupt controller architecture

16 interrupts of each channel, including null interrupts (no interrupt request in current channel)， are written. A new interrupt of a certain channel can not be generated unti1 the previous ones are responded so 16× 16=256 bits of depth are required at 1east to store the interrupts Implementation of the controller is shown in Fig. 10. The conclusion is that the time of null interrupts output is the main time overhead.

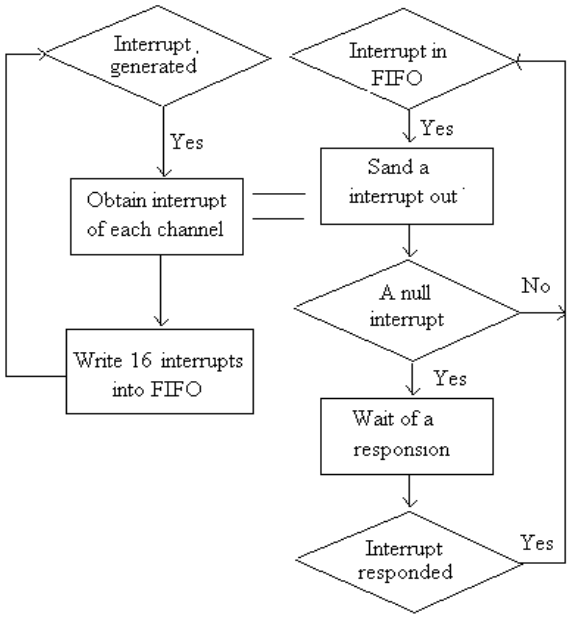

Fig. 10 Interrupt controller implementation

But the system main clock frequency is far faster than interrupt generating frequency so the overhead can be ignore.

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.

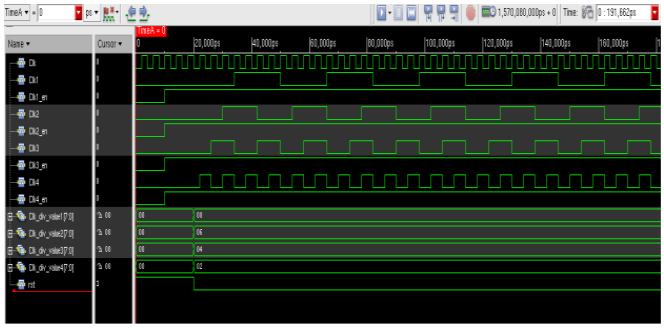

Fig 11: Baud Rate Generator Output

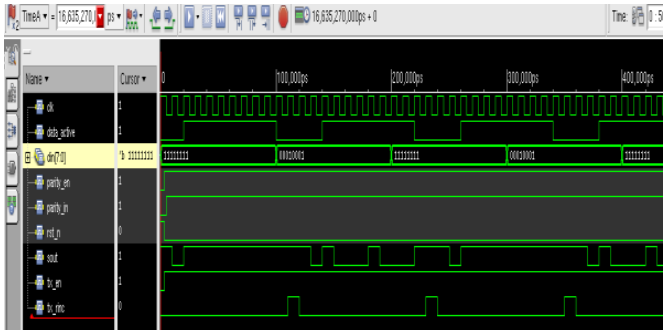

Fig 9: Transmitting Sequence.

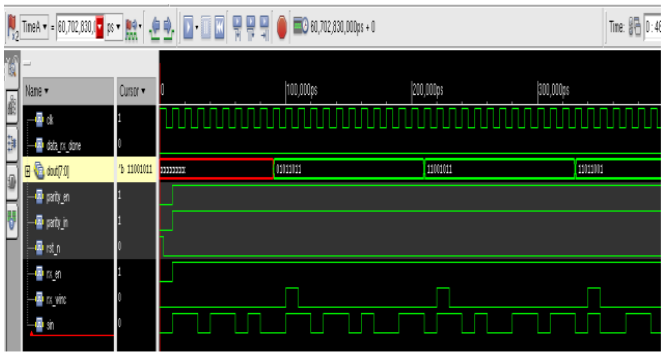

Fig 12: Receiver Response

V. CONCLUTION

The paper presents design method of asynchronous FIFO and structure of the controller. This controller is designed with FIFO circuit block and UART (Universal Asynchronous Receiver Transmitter) circuit block within FPGA to implement communication in modern complex control systems quickly and effectively. Form the communication sequence diagrams; it is easily to know that this controller can be used to implement communication when master equipment and slaver equipment are set at different Baud Rate. It also can be used to reduce synchronization error between sub-systems in a system with several sub-systems. The controller is reconfigurable and scalable.

REFERENCES

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

2. Free scale semiconductor, Inc MPC860 Power QUICCTM family user's manua1．2004．

3. Li SG Gao D Y Nie PQ Study on multi task management unit MTU of embedded micro NCS, Acta Aeronautical et Astronautic Sinica 2000；21(2)：134-137 in Chinese.

4. Liu L, Gao D, Y Zhang SB, et a1. Design of EM FPU in embedded microprocessor Act a Aeronatuticaet Astronautics Sinica2001；22(4)：308-31 1．[in Chinese]

5. Yean del J, Thulborn D, Jones S. An online testable UART implemented using IFIS．1 5th IEEE VLSI Test et Astronautic Symposium，1997；344-349．

6. Elmenreich W, Delvai M, Time triggered communication with UARTs．4th IEEE International Workshop on Factory Communication Systems，2002；97-104．

7. Gallo R, Delvai M，Elmenreich W，eta1．Revision an d veil fiction of an enhanced UART．IEEE International Workshop on Factory Communication Systems，2004；315-318．

8. Delvai M，Eisenmann U, Elmenrichs W Intelligent UART module for real-time applications. First Workshop on Intelligent Solutions in Embedded Systems，2002；1 77—185．

9. X., Yang, "Industrial Data Communication and Control Networks", Beijing: TUP, 2003.6[9] B. Zeidman, "Designing with FPGAs & CPLDs", CMP Books, 2002

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