Implementation of a Multi-channel UART Controller Based on FIFO Technique and FPGA

Shouqian Yu Lili Yi Weihai Chen Zhaojin Wen
School of Automation Science and Electrical Engineering
Beijing University of Aeronautics & Astronautics
Beijing 100083, China
ysq@buaa.edu.cn buaa_lily@126.com whchenbuaa@126.com junwen1981@126.com

Abstract: **To meet modern complex control systems communication demands, the paper presents a multi-channel UART controller based on FIFO(First In First Out) technique and FPGA(Field Programmable Gate Array). 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.**

Keywords: **FIFO, FPGA, UART**

I INTRODUCTION

Today, owing to availability of state-of-the-art microcontrollers and digital signal processors (DSPs), complex control algorithms can be easily implemented to attain the desired system performance. But 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 [1]. 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.

In several control systems, 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. 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].

In some complex systems, communications between the master controller and slaver controllers are implemented by serial or parallel port. 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. But sometimes a common serial port could not meet requirements of complex systems with different Baud Rates equipments even some special Baud Rate equipments. As showing in figure 1, 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.

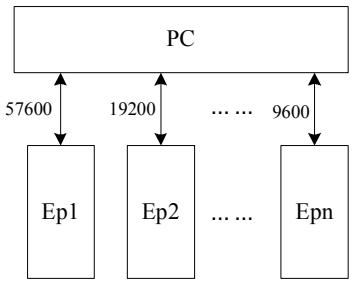

Fig.1. Multi-equipments communication diagram

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.

To solve these problems described as above, we design a multi-channel UART controller based on FIFO techniques and FPGAs. 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.

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. In this paper, using a FPGA-EP1C6Q produced by ALTERA and FIFO techniques design a Baud Rate converter 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.

In several systems such as high data collection system, high speed control system based on PCI and multi-DSP signal processing system, FIFO is used to complete communication between high speed device and low speed device or to complete communication between the same sub-controller. FIFO is the most important part of these systems and it works as a bridge between different devices [3, 4]. As the same, in our controller, asynchronous FIFO based on FPGA is also the most important part. 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 DESIGN OF ASYNCHRONOUS FIFOS

A. *Introduction to 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, where the two clock domains are asynchronous to each other. FIFOs are always used for data cache, storing differences of frequency or phase of asynchronous signals. And 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 is showing in figure 2.

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 can not accommodate more data and if the "Empty" is active then the FIFO can not 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.

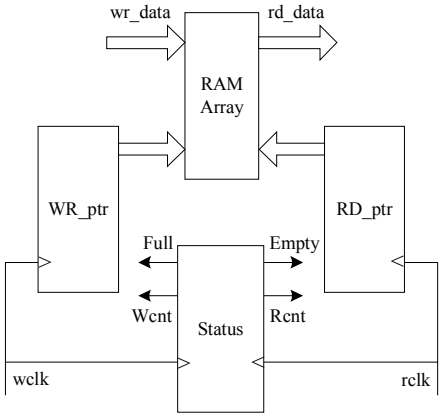

Fig.2. Asynchronous FIFO structure diagram

In designing of asynchronous FIFOs, two difficult problems can not 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 can not 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 [5].

In order to exactly know weather 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".

In this paper, this method is instead of another design technique used to distinguish between full and empty is to add an extra bit to each pointer. The pointers length n equals to $\log_2(array\_size)$ .The $array\_size$ is the depth of the FIFO needed in a project. For example, when setting the $array\_size$ of the FIFO 64 byte (8 bits one byte), the writer and read pointers' length is $\log_2 64 = 6$. Using $n+1$ bits pointer when n is the number of address bits required to access the entire FIFO memory buffer. when both pointers including the MSBs are equal the FIFO is empty. And the FIFO is full when both pointers , except the MSBs are equal. As figure 3 showing, when the $array\_size$ of the FIFO is 8 bytes, the number of address bits required to access the entire FIFO memory buffer is 4. With an additional address bit the full and empty signals can be easily created [5, 6].

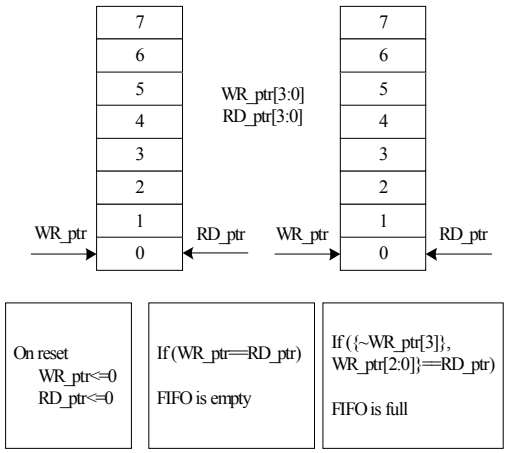

Fig.3.FIFO full and empty conditions

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. Metastability 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. Unpredictability also implies another phenomenon and this is the real danger that metastability poses. To eliminate these kinds of problems caused by metastability is a difficulty in designing a FIFO [5, 7].

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 $t = 0$, 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 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. The conversion between the Binary codes and the Gray codes is as following:

$$\begin{aligned} g_n &= b_n \\ g_i &= b_i \oplus b_{i+1} \forall i \neq n \end{aligned} \tag{1}$$

and

$$\begin{aligned} b_n &= g_n \\ b_i &= g_i \oplus b_{i+1} \forall i \neq n \end{aligned} \tag{2}$$

There are multiple ways to design a Gray code counter and this paper details a simple and straight forward method to do the design. The technique describe in this paper uses only one set of flip-flops for the Gray code counter as showing

in figure 4. 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.

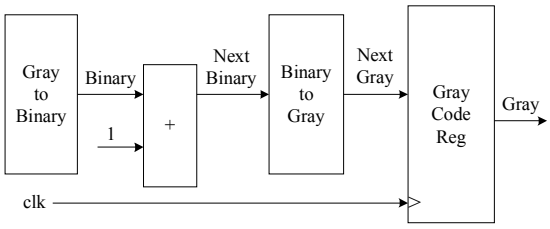

Fig.4.Gray counter architecture

III IMPLEMENTATION OF A MULTI-CHANNEL UART CONTROLLER

A. Hardware structure

In the multi-channel controller, there are different blocks including UART block, Status Detectors, asynchronous FIFOs block and Baud Rate Generator block. Each block has different function in the controller.

The first part is UART circuit block and its structure is shown in figure 5. 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 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.

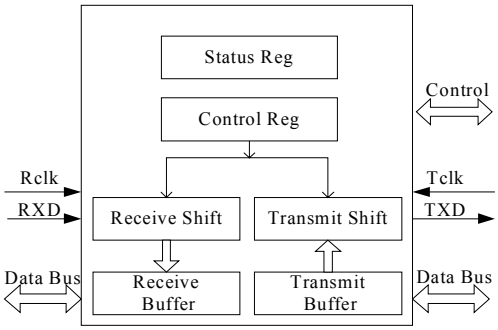

Fig.5.Structure of UART block

Using FIFO technique and the COM block as mentioned before, we design a multi-channel controller. It can be used to implement communications between MCUs in a complex system. And it can also be used to complete communication between high speed device and low speed device. Structure of the controller is showing in figure 6. The controller is built within a FPGA – EP1C6Q240 which is based on SRAM technique produced by ALTERA. It is possible to design small scale memorizer like FIFOs. When designing FIFOs within FPGAs, you should consider the capacity of FIFO in practice and also consider the FPGAs' capacities.

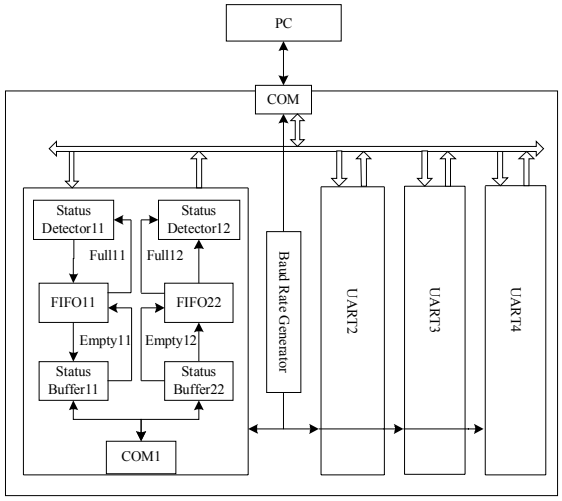

Fig. 6. Structure of the controller

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. 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. The software flow chart is shown in figure 7.

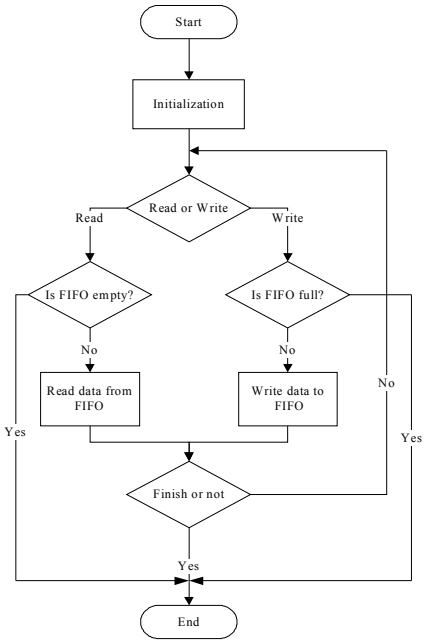

Fig.7. The Software flow chart

As showing in figure 7, when FIFO is full you can not 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 can not 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 Modelsim. 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. As showing in figure 8, 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 figure 9. 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 figure 9. All sub-MCU can receive data at the same time.

We design four channels in Verilog HDL as totally the same structure use *always* block to implement communication. So on theory, 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 causes sub-controllers can not 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 [8, 9].

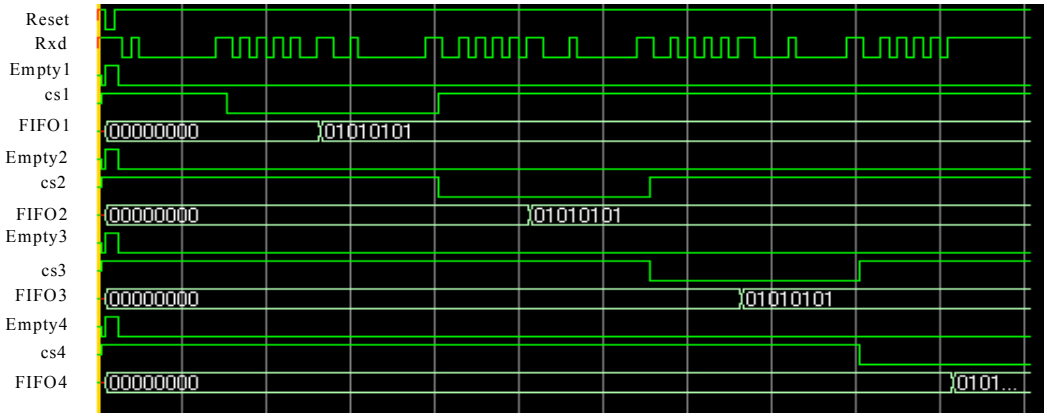

Fig.8. Receiving sequence

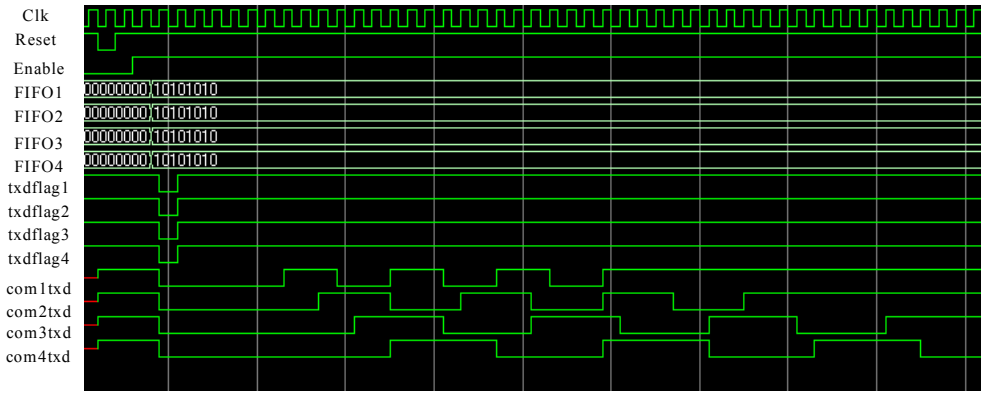

Fig.9. Transmitting sequence produced by different Baud Rates

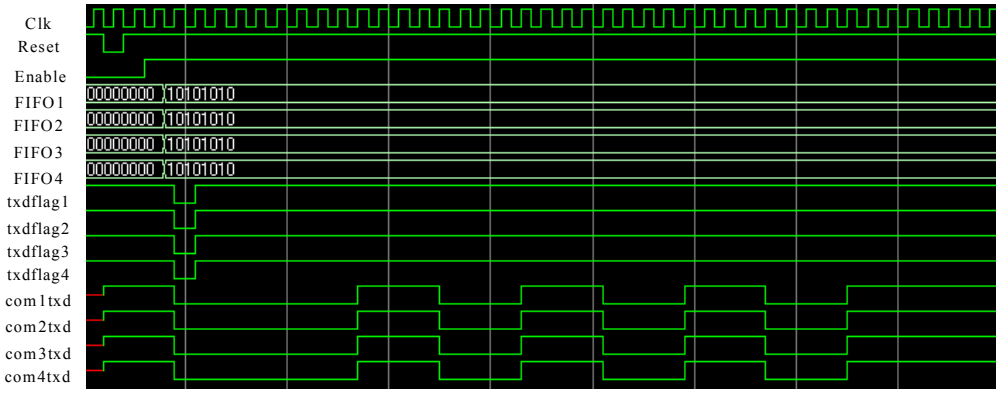

Fig.10. Transmitting sequence produced by the same Baud Rate

V CONCLUTION

This paper introduces a method to design a synchronous FIFO based on FPGA. And using synchronous FIFO technique implements a multi-channel UART controller within FPGA based on SRAM with high speed and high reliability. 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. The fault of FPGA based on SRAM is reconfigurable, so the controller's fault is that it would be influenced by the radiation from surroundings and by short-time pulses easily.

ACKNOWLEDGMENT

This work is supported by natural science foundation of China under the research project 60575052 and 50375008.

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 Petro-chemical 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