TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 12/18 pp812-816
Volume 13, Number 6, December 2008

Low Latency High Throughout Circular Asynchronous FIFO*

XIAO Yong (肖 勇), ZHOU Runde (周润德)**

Institute of Microelectronics, Tsinghua University, Beijing 100084, China

Abstract: This paper describes a circular first in first out (FIFO) and its protocols which have a very low latency while still maintaining high throughput. Unlike the existing serial FIFOs based on asynchronous micropipelines, this FIFO's cells communicate directly with the input and output ports through a common bus, which effectively eliminates the data movement from the input port to the output port, thereby reducing the latency and the power consumption. Furthermore, the latency does not increase with the number of FIFO stages. Single-track asynchronous protocols are used to simplify the FIFO controller design, with only three C-gates needed in each cell controller, which substantially reduces the area. Simulations with the TSMC 0.25 μm CMOS logic process show that the latency of the 4-stage FIFO is less than 581 ps and the throughput is higher than 2.2 GHz.

Key words: asynchronous circuit; asynchronous first in first out (FIFO); circular; systems on a chip (SOC); global asynchronous local synchronous (GALS)

Introduction

As systems on a chip (SOC) become larger and faster, it is becoming increasingly difficult to distribute a single synchronous clock to the entire chip[1,2]. Any communication crossing two asynchronous clock domains requires careful synchronization; and the first in first out (FIFO) working in asynchronous clock domains is well competent for the job[3]. However，most existing FIFOs have high throughput at the cost of high forward and reverse latency. The delay from the input to the output in an empty FIFO is defined as the forward latency and the delay from the output to the input in a full FIFO is defined as the reverse latency. The existing FIFOs working in asynchronous clock domains can be designed with synchronous circuits and with asynchronous circuits.

In FIFOs using synchronous circuits, the computation of the empty-flag requires three steps. The first step is generally to transfer the address of the input to Gray-codes which is then latched by the clock of the input before being sent to the output. In the second step, the Gray-code address is latched at the output port through the synchronizer to ensure that the probability of a metastable state occurring is extremely small. In the third step, the input and the output addresses are compared to generate the empty-flag after the Gray-code input address is transferred back to binary code. Thus, the delay from the input to the output can be expressed by $T_d = T_{clk1}+T_{clk2}+T_{G2B}+T_{rr}$, where T_{clk1} is the input clock cycle, T_{clk2} is output clock cycle, T_{G2B} is the delay for transferring the Gray-code to the binary code, and T_{rr} is the delay between the positive edges of the input and the output clocks. Therefore, the forward latency of this type of FIFO will be longer than T_d and the delay for transferring the Gray-code to the binary code will also affect the read cycles.

The other way to implement FIFO working in asynchronous clock domains is to use asynchronous circuits.

Received: 2007-01-08; revised 2007-05-25

* Supported by the National Key Basic Research and Development (973) Program of China (No. 2006CB302700) and the National High-Tech Research and Development (863) Program of China (No. 2007AA01Z2B3)

** To whom correspondence should be addressed.
E-mail: zhourd@mail.tsinghua.edu.cn; Tel: 86-10-62787292

Most existing asynchronous FIFOs use a serial architecture to trade higher latency for higher throughput[4-6]. In the serial architecture, data must be written into the input cell and then read from the output cell, with the input data passed through all the cells to reach the output port, so the latency increases linearly with the number of FIFO stages. However, practical applications such as global asynchronous local synchronous (GALS) systems want a low latency for data items[7-11].

This paper describes a FIFO with a circular architecture and its associated protocols to provide high throughput with low latency. In this architecture, each cell is connected to the input and output ports directly by a common bus. Once a data item is written into the FIFO, only one relevant cell will respond to the operation and latch the data. Thus, the input data item is immediately available for the output with no need to pass through all the intermediate cells. Therefore, this circular architecture gives a low latency that is almost independent of the number of FIFO stages.

1 Architecture and Protocol

In an asynchronous FIFO, the first data into the input port will be the first out of the output, with the data timing then related to the data output time. A key difficulty in the design is to avoid metastable states.

1.1 Architecture

The FIFO circular architecture is shown in Fig. 1, where each FIFO cell is composed of a controller and a storage unit. The controller performs handshaking with the input/output interfaces and generates the corresponding wr and rd signals. Each storage unit stores or outputs data when the wr or the rd signal is active. When an input event occurs, only one cell, pointed to by the write pointer, will respond to the event, and the wr signal of that cell will be active so that the data is stored in that cell. The read process is similar.

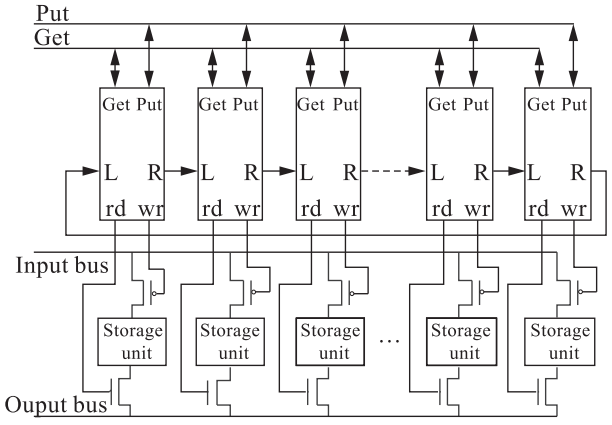

Fig. 1 Proposed circular asynchronous FIFO

1.2 Write and read operation protocols

A cell's state is 1 if it has been written to, and a cell's state is 0 if it has been read. Assume that at the beginning, the FIFO is empty and all cell states are 0. The write and read protocols are described as follows:

Protocol 1 When a cell's state is 1 and its left cell's state is 0, the read pointer points to this cell;

Protocol 2 When a cell's state is 0 and its left cell's state is 1, the write pointer points to this cell;

Protocol 3 When read operation occurs, the cell pointed to by the read pointer will be read, and the cell's state will be set to 0;

Protocol 4 When write operation occurs, the cell pointed to by the write pointer will be written, and the cell's state will be set to 1.

Figure 2 gives an example of write and read operations, with the cell at the top position of the circle being defined as cell0, and other cells referred to as cell1 to cell7 in the clockwise direction. At the initial state, cell7 satisfies the condition of protocol 1 and cell3 satisfies the condition of protocol 2, as shown in Fig. 2a, so the read pointer points to cell7 and the write pointer points to cell3. After a write operation, the data is written into cell3 and the state of cell3 becomes 1. At this time, cell4 satisfies the condition of protocol 2, so the write pointer points to cell4 as shown in Fig. 2b. Similarly, after a read operation, the data is read from cell7 and the state of cell7 becomes 0, and the FIFO's states are shown in Fig. 2c.

One problem with these protocols is that when the

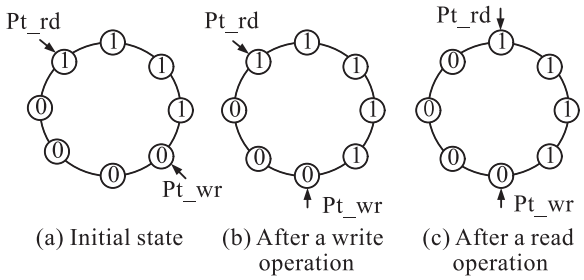

Fig. 2 An example of write and read operations

FIFO is empty or full, no cells satisfy the condition of protocol 1 or protocol 2. This problem, however, can be solved if information of the write and read pointers is saved before the FIFO becomes full or empty, as part of the cell's protocols.

1.3 Cell's asynchronous handshake protocol

All the signals in the circular FIFO are shown in Fig. 1, where R is the state of the current cell and L is the state of its left-hand cell. Besides these interface signals, two other signals, Pt_wr and Pt_rd, are used to present whether the cell is pointed to by the write or read pointers. All the signal transitions are shown in Fig. 3. Each cell has three handshaking processes, handshaking with both the input and output interfaces, and handshaking with the two neighbor cells to exchange their state information since state information changes of two neighbor cells are asynchronous. A single-track asynchronous protocol[12-14] is used to implement the first two handshaking processes. Figure 3 presents the beginning of a write process when Put has been driven to low. At this time, only one cell whose Pt_wr is active will respond to the event and drive wr to low to store the data. After the storing is completed, the cell will drive Put to high to indicate that a new write process can be initiated, with wr simultaneously driven to high. The read process is similar, with the active levels of all signals in the read process inverted with respect to the write process to simplify the circuit design.

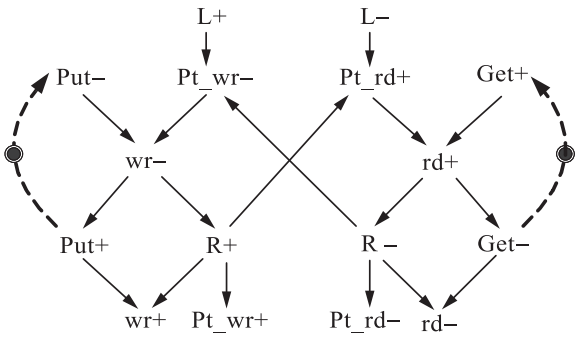

Fig. 3 Signal transition graph of the circular FIFO

However, it has different scenarios when the entire FIFO is empty or full. Assume that only one cell's state is 1 in the whole FIFO, which is referred to as cell *k*. In this case, the Pt_rd of cell *k* and the Pt_wr of cell *k*+1 are active (the Pt_rd of cell *k* is high and the Pt_wr of cell *k*+1 is low). When cell *k* has been read before cell *k*+1 is written, the FIFO will become empty. At this time, the Pt_rd of cell *k* becomes low and there is no cell whose Pt_rd is active. Therefore, if a read operation occurs, no cell will respond. Notice that after cell *k* has been read, the L of cell *k*+1 became low, but the Pt_rd of cell *k*+1 was still active. Thus, if a write operation occurs at that time, data will be written into cell *k*+1. When the FIFO is full, if the left cell has been written, the read pointer of the current cell will remain active until it has been read. Therefore, whether the FIFO is empty or full, the FIFO's write and read operations will function correctly.

2 Circuit Design and Performance Analyses

To simplify the circuit design, the two signals Pt_wr and Pt_rd are merged into one signal, Pt, in the signal transition graph. In this way, two asymmetric C-gates can be replaced with a single symmetric C-gate in the circuit design. However, a problem may arise when the two signals are merged into one, since after a cell has been written, the Pt will not be driven to high immediately until its left cell has been read. Therefore, to avoid the cell being written repeatedly before it is read out, the negative transition of wr may take place only when Put, Pt, and R signals are all low. The read process is similar. The simplest merged signal transition graph (STG) is shown in Fig. 4.

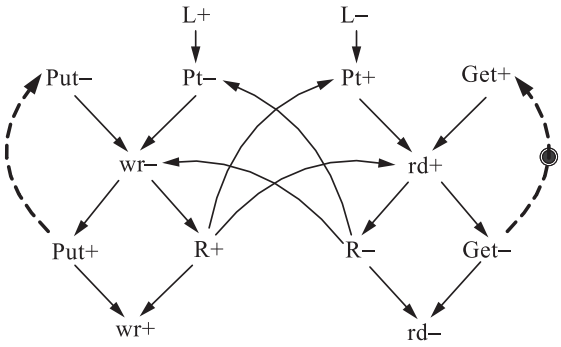

Fig. 4 Merged signal transition graph

2.1 Circuit design

The necessary condition for wr becoming low is that Pt, R, and Put are all low and the necessary condition for wr becoming high is that R and Get are both high. An asymmetric C-gate cell can be used to implement this logic. The detailed circuit implementation of the asymmetric C-gate was described by Furber and

Day[15]. Similarly, the logic for generating rd signal can also use an asymmetric C-gate.

The circuit in Fig. 5 shows two cell implementation of the merged STG. After reset, the FIFO is empty, the Pt of cell0 is cleared and the Pts of other cells are all set. When the first FIFO's write operation occurs, cell0 will be written, so cell0's symmetric C-gate is connected to Clear and all other cells (e.g., cell1) are connected to Set. Both Put and R are driven to high by the signal wr. This can be easily designed such that the delay between the positive edge of Put and the positive edge of R is less than the delay sum of the C-gate and the Inverter. The Put becomes high again before the R of cell0 drives the Pt of cell1 to low; therefore, the design easily guarantees that only one cell will be written once a write operation has been issued. The read operation is similar to the write operation.

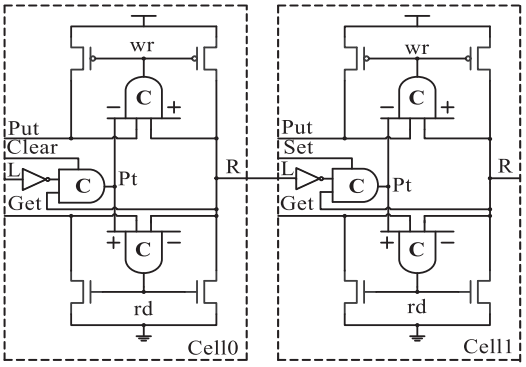

Fig. 5 Two cells' circuit implementation for the FIFO signal transition

2.2 Performance analyses

If one signal transition requires one gate delay, then one C-gate delay is equivalent to two gate delays. A write operation begins with Put being driven to low. After the written data has been latched, the cell will drive Put to high to indicate that new data may now be sent to it from outside. Thus, the delay from Put becoming low to Put returning to high is three gate delays. Assume that the system also needs three gate delays to set Put low again after detecting that Put is high. Then it needs six gate delays to complete a write operation. A read cycle also requires six gate delays.

When the FIFO is empty, the system performance is determined by the FIFO's forward latency, which is defined as the delay from Put to Get. As shown in Fig. 5, the critical path is from Put to wr, R, Pt, rd, and Get; therefore, the forward latency is eight gate delays. The critical path passes through only one cell in this circular FIFO, so the latency of this circular FIFO is independent of the number of FIFO stages. In serial FIFOs, the critical path is from the input cell to the output cell, passing through all intermediate cells. Similarly, when the FIFO is full, the reverse latency determines the system's performance, with the reverse latency also being eight gate delays.

3 Simulation Results

The FIFO was analyzed in HSPICE using the TSMC 0.25 μm CMOS logic process (V_{dd}=2.5 V, T=25℃) parameters.

Figure 6 shows the HSPICE simulation waveforms for the 4 stages circular FIFO, with the FIFO functioning correctly for all conditions. The FIFO is initially empty. When the Get signal becomes high to read data, the FIFO will not respond until the first write operation has been completed. The forward latency is obtained by measuring the delay between the first falling edge of Put and the first falling edge of Get. After five write operations, the FIFO is full, so the sixth write operation is pending until the cell1 data has been read. The delay between the second rising edge of Get and the sixth rising edge of Put is the reverse latency.

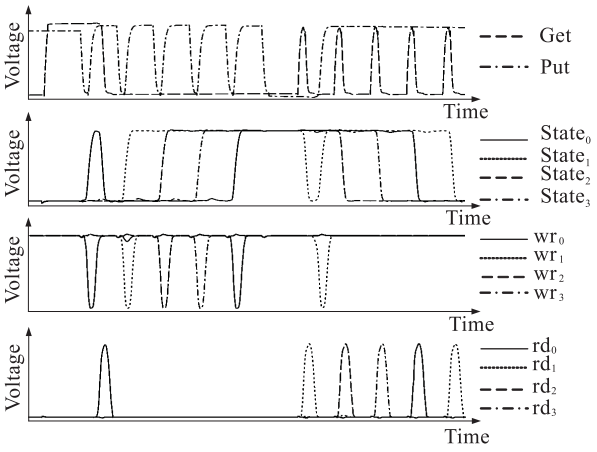

Fig. 6 Four-stage FIFO simulated waveforms

The simulation results for circular FIFOs with 4 to 8 stages are shown in Fig. 7. The 4-stage circular FIFO can work correctly up to 2.2 GHz, with a latency of less than 581 ps. When the number of circular FIFO stages is doubled (from 4 to 8), the latency increases slightly by 6.9%. When the number of stages increases from 4 to 16, the latency increases by 16.8% since the MOSFETs used in the FIFOs with 4 to 16 stages have

the same sizes. If the MOSFETs' sizes for FIFOs with different numbers of stages were optimized, the latency would not increase much with the number of stages. Unlike existing serial FIFOs based on asynchronous micropipelines, the simulation results show that the latency of the circular FIFO is almost independent of the number of stages, while the throughput still maintains high. The effect of the increased number of stages on the latency is 50% less than for the bus-based FIFO by Chelceq and Nowick[16].

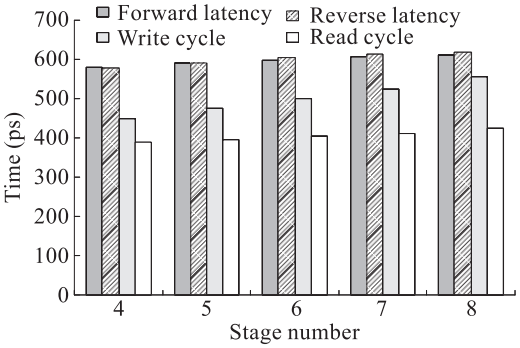

Fig. 7 Simulated results for circular FIFOs

4 Conclusions

This paper describes and analyzes a circular architecture FIFO to deliver a new type of FIFO suitable for GALS systems. Simulations show that the circular FIFO has a very low latency and a high throughput. Compared with serial FIFOs based on asynchronous micropipelines, the circular FIFO latency does not increase with the number of FIFO stages and can reduce power consumption during data transfer. The use of the single-track asynchronous protocol greatly simplifies the circuit design with only three C-gates needed in each cell controller, which significantly reduces the chip area.

References

[1] Friedman E G. Clock distribution networks in synchronous digital integrated circuits. *Proceedings of the IEEE*, 2001, **89**(5): 665-692.

[2] Martin A J, Nystrom M. Asynchronous techniques for system-on-chip design. *Proceedings of the IEEE*, 2006, **94**(6): 1089-1120.

[3] Chelcea T, Nowick S M. Robust interfaces for mixed-timing systems. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2004, **12**(8): 857-873.

[4] Molnar C E, Jones I W, Coates W S, et al. Two FIFO ring performance experiments. *Proceedings of the IEEE*, 1999, **87**(2): 297-307.

[5] Lee J G, Kim S J, Lee J A. Handshake-wave combined approach with runtime reconfiguration for designing a low latency asynchronous FIFO. In: Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits. Fukuoka, Japan, 2004: 188-199.

[6] Sutherland I, Fairbanks S. GasP: A minimal FIFO control. In: Proceedings of 7th International Symposium on Asynchronous Circuits and Systems. Salt Lake City, USA, 2001: 46-53.

[7] Dobkin R, Ginosar R, Sotiriou C P. Data synchronization issues in GALS SoCs. In: Proceedings of 10th International Symposium on Asynchronous Circuits and Systems. Crete, Greece, 2004: 170-179.

[8] Dobkin R, Ginosar R, Sotiriou C P. High rate data synchronization in GALS SoCs. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 2006, **14**(10): 1063-1074.

[9] Moore S, Taylor G, Mullins R, et al. Point to point GALS interconnect. In: Proceedings of 8th International Symposium on Asynchronous Circuits and Systems. Manchester, UK, 2002: 69-75.

[10] Mekie J, Chakraborty S, Venkataramani G, et al. Interface design for rationally clocked GALS systems. In: Proceedings of 12th IEEE International Symposium on Asynchronous Circuits and Systems. Washington DC, USA, 2006: 160-171.

[11] Muttersbach J, Villiger T, Fichtner W. Practical design of globally-asynchronous locally-synchronous systems. In: Proceedings of 6th International Symposium on Asynchronous Circuits and Systems. Eilat, Israel, 2000: 52-61.

[12] Xiao Yong, Zhou Runde. Single-track asynchronous pipeline controller design. In: Proceedings of the ASP-DAC 2005. Shanghai, China, 2005: 764-768.

[13] Ferretti M, Beerel P A. Single-track asynchronous pipeline templates using 1-of-*N* encoding. In: Proceedings of Design, Automation and Test in Europe Conference and Exhibition. Paris, France, 2002: 1008-1015.

[14] Ozdag R O, Beerel P A. High-speed QDI asynchronous pipelines. In: Proceedings of 8th International Symposium on Asynchronous Circuits and Systems. Manchester, UK, 2002: 12-22.

[15] Furber S B, Day P. Four-phase micropipeline latch control circuits. *IEEE Transactions on Very Large Scale Integration* (*VLSI*) *Systems*, 1996, **4**(2): 247-253.

[16] Chelceq T, Nowick S M. Low-latency asynchronous FIFO's using token rings. In: Proceedings of 6th International Symposium on Asynchronous Circuits and Systems. Eilat, Israel, 2000: 210-220.