Design and implementation of elastic buffer for Universal Serial Bus 3.0

Hong Chang[1a)], Te Peng[1], Daoming Ke[1b)], Tailong Xu[2], Jian Meng[1]

1 School of Electronics and Information Engineering, Anhui University, Hefei, 230601, China

2 Department of Electronic Information and Electrical Engineering, Hefei University, Hefei.230601, China

a) 814874822@qq.com

b) kedaoming@sohu.com, Corresponding Author

Abstract: To avoid the timing errors caused by the asynchronous control of elastic buffer between read and write operations. In this paper, a new skip (SKP) adding technology is proposed, which is achieved by the suspending of the read pointer. Based on the proposed new mechanism, a new elastic buffer is designed and realized under the TSMC65nm CMOS technology. The post simulation results show that, the presented elastic buffer can complete SKP adding and removing function correctly, and the clock frequencies can reach 500 MHz with tiny changes in area, energy consumption and speed of reading. It can be applied to the design and optimization of USB3.0.

Keywords: USB3.0, Elastic buffer, Read pointer, SKP

Classification: Integrated Circuits

References

[1] Seong. Jae. Jeong and Keum. Cheol. Hwang: IEICE Electron. Express **7** (2010) 1370

[2] Szecowka. P. M and Pyrzynski. K. J: ICSES Tech. Dig. (2012) 1

[3] Wikipedia: Universal serial bus (2011) https://en.wikipedia.org/wiki/USB_3.0

[4] Intel Corporation, Microsoft Corporation, NEC Corporation, et al. [S]. USA: USB Organization, 2008, 11.

[5] Nikolaos. Terzopoulos, Costas. Laoudias, Fotis. Plessas: International Journal of Circuit Theory and Application **43** (2015) 900

[6] Kopanski. J, Pleskacz. W. A and Pienkowshi. D: DDECS Tech. Dig. (2011) 131

[7] Joe. Winkles, "Elastic Buffer Implementation in PCI Express Devices". Mindshare Inc, pp: 1-16. November, 2003.

[8] Intel Corporation: PHY interface for the PCI Express, SATA, and USB3.0 architectures (2013) http://www.intel.cn/content/www/cn/zh/io/pci-express/phy-interface-pci-express-sata-usb30-architectures.html?wapkw=phy+interface+fo

DOI: 10.1587/elex.13.20151084
Received December 18, 2015
Accepted January 25, 2016
Publicized February 12, 2016

r+the+pci+express,+sata,+and+usb3.0+architectures&_ga=1.73465044.77706
7156.1444875981.

[9] Xiao jian, Chen guican, Zhang fujia: Journal of Semiconductors **29** (2008) 1417

[10] David Xing: Sciencepaper Online. Papers. (2012) 1.

[11] Zhu xiaoming, Wang xiaoli and Cheng zeng: MICROELECTRONICS & COMPUTER **29** (2012) 117.

[12] Michelogiannakis G. Balfour and J. Dally. W. J: IEEE Transaction **62** (2013) 295.

[13] Cumming. C. E and Alfke. P: simulation and synthesis techniques for asynchronous FIFO design, SNUG San Jose, (2002).

[14] Zheng zhengbing: Journal of Computer Applications **32** (2012) 3259

1 Introduction

With the development of Universal Serial Bus (USB) technology, the traditional series and data interface technology have long since been replaced [1,2]. As a new interface technology, Universal Serial Bus 3.0 (USB3.0) has been used in the PC field for recent years [3], which was released by Intel, Microsoft, NEC and other companies in November, 2008 [4]. Based on the USB2.0, it adds a new SuperSpeed transfer channel, which promotes the data rate ten times higher than before to 5 Gbps [5]. However，it still belongs to the transport protocol of high speed, serial and source synchronous transmission, relative to USB2.0, which transmitting serial data in the sender in the form of differential signal and deriving serial data and clock by the Clock Data Recovery (CDR) circuit in the receiver. Because the sending and receiving ends adopt independent reference clock sources, a certain frequency difference can arise between the clock frequency restored from the ends of the receiving and the local one [6]. In order to ensure the correctness of data transmission and the synchronization of the data of clock domain, which recovered from the clock data recovery circuit to the local clock domain, USB3.0 uses the elastic buffer to compensate the clock frequency and achieve phase synchronization [7].

Elastic buffer was earliest put forward by Maurice Karnaugh in Pulse Code Modulation (PCM) signal for the telephone network transmission, which achieves the goal of the clock compensation by adding or removing specific symbols [8]. In the circuit design of USB 3.0, to compensate the clock frequency difference between the two clock domains, the sender sends a sequence of SKP at an average of every 354 characters and the elastic buffer delete or add SKP symbol according to the instruction [4].

In conclusion, for the integrity of the data, the design for elastic buffer has become an important link in USB 3.0 [9]. Many scholars have researched the design of the elastic buffer, such as Dvaid Xing [10] and Zhu Xiaoming [11] who put forward the method of using breakpoint restore, pointer jump and write pointer suspend to achieve the design of elastic buffer. However, a complex asynchronous control logic circuit is need by this method and the timing errors may appear during the process of adding SKP symbol. In this paper, in order to avoid this problem, a new method of adding SKP symbol by the suspending of the read

pointer is proposed for the design of elastic buffer, which make the process of adding and deleting are finished in the two clock domains, respectively. In this way, the timing errors caused by asynchronous control can be avoided. In addition, it is no need to write pointer jump and breakpoint preserve in the process, which reduce the utilization of logic units and registers. Under the TSMC65nm process condition, this paper has accomplished a new elastic buffer and made emulation proof to it. The post-simulation results show that in the more conservative temporal constraints, both the read and write clock frequencies of this newly-introduced elastic buffer can reach 500 MHz.

The rest of this article is organized as follows. Section 2 of this paper describes the design principle of elastic buffer. Section 3 introduces implementation of the proposed elastic buffer and section 4 shows the simulation results of the design.

2 Proposed principle of design

2.1 Methodology

Generally, there are two realizing methods for elastic buffer: the flow control method and the normal half-full method. The former is to maintain the elastic buffer in bottom entry point state [12]. Although this implementation method doesn't require more depth of elastic buffer, more complex additional circuit should be added to generate the corresponding enable signals. Moreover, because of the control of enable signal for subsequent clock circuit, it produces the gated clock, which is harmful for the realization of Design for TEST (DFT). The normal half-full method is used to maintain elastic buffer in half full state [8]. Compared to the method of flow control, it does not require more complex additional circuit which is conducive for the realization of the DFT though the depth of the elastic buffer increased. Therefore, the method of normal half-full is primary concerned of this paper.

2.2 Capacity

According to the USB3.0 specification, the total magnitude of the frequency delta between the system clock and receive clock can range from -5300ppm to 300ppm [4]. In addition, the maximum size of the data packet is 1056 bytes and SKP symbol should not be inserted within any packet, thus the maximum data displacement is given as follows,

$$maximum \quad data \quad offset = \frac{300-(-5300)}{1000000} * 1056 \approx 8$$

According to the shift of maximum symbols and the implementation of normal half-full, the EB's buffer capacity will be set to 16 in this paper.

2.3 The proposed working principle

The nature of elastic buffer is an asynchronous First In First Out（FIFO）where data is deposited at a certain rate based on one clock and removed at a rate derived from the other [13,14]. As shown in Fig.1， in most cases, the state of elastic buffer will remain in half full while its valid data size will maintain on 8 due to the same frequency of read and write clock.

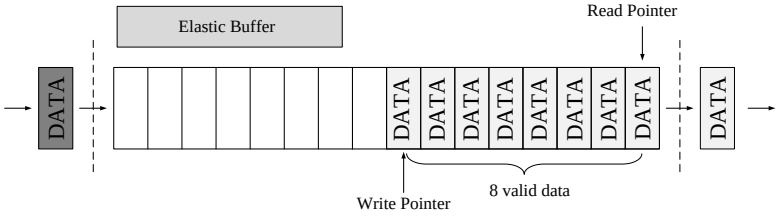

Fig. 1. Read and write operations in the same clock frequency

If the frequency of the read clock is higher than that of the write clock, the amount of read data will be more than that of write one, which even causes a decrease in the number of valid data until the underflow of the FIFO will be exist. In order to avoid such situation, the effective data volume of elastic buffer is increased by adding SKP symbol to keep it in a half-full state. In this paper, a new approach for implement SKP adding by the suspending of the read pointer is presented. As shown in Fig. 2, when the available SKP instruction appears, the elastic buffer will suspend the read pointer and add SKP symbol to the output data to ensure its half-full state.

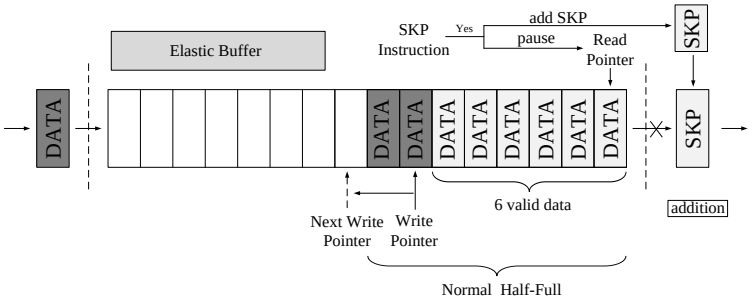

Fig. 2. Operations under the write clock higher than the read clock

On the contrary, the amount of write data will be more than that of read one, which causes an increase in the number of valid data and even to the overflow in the FIFO. As shown in Fig. 3, when the deleting command is effective, the elastic buffer should suspend the write pointer and allow the next data to cover the data of the current address to complete the deletion of SKP symbol. In this process, the data stored in FIFO can be read out normally, which guarantees the state of elastic buffer at half-full by reducing the volume of effective data.

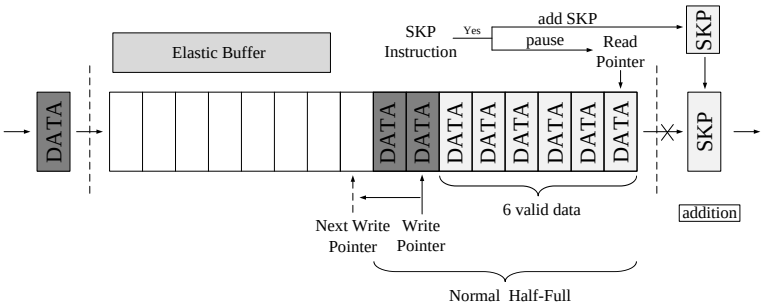

Fig. 3. Operations under the write clock higher than the read clock

3 Implementation of elastic buffer

As shown in Fig. 4, the new elastic buffer have seven units with two clock domains: (a) the Local Clock Domain is used to clock all of its internal gates and to transmit data, and (b) the Recovered Clock Domain is used to latch inbound data.

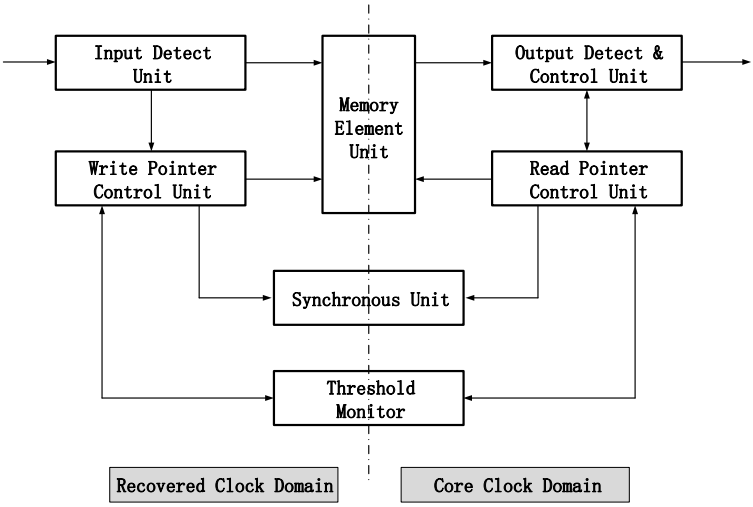

Fig. 4. The architecture of Normal Half-Full elastic buffer

The function of input detect unit is to detect whether the input data is SKP, and an instruction for SKP deleting operation will be provided according to the testing results.

Binary and gray codes of the write pointer are generated by the write pointer control unit. The binary code of write pointer is used for the selection of the storage unit of data writing, and generates the instruction for SKP deleting operation when compared with the read pointer after being synchronized. Likewise, the gray code of write pointer **is** synchronized to the local clock domain and compared with gray code of read pointer then the unit generates a null flag.

The role of the read pointer control is to produce the binary code and gray code of read pointer. The binary code of read pointer is the address of the storage unit of data reading and it generates the request sign of SKP add after being compared with the synchronized write pointer. The gray code of read pointer is synchronized to the Recovered Clock Domain and compared with the gray code of the write pointer,then the unit produce full flag.

The output detect & control unit determines whether the data read from the storage unit is the SKP character or the END character, and then generates the corresponding symbol. Besides, it also puts out data according to the result of detecting.

The function of synchronous unit is to synchronize the gray code of write pointer and read pointer to the Local Clock Domain and Recovered Clock Domain, respectively. When the difference between the gray code of read pointer and the synchronized gray code of write pointer is larger than 16, it produces empty sign in the Local Clock Domain. Similarly, when the difference between the gray code of write pointer and the synchronized gray code of read pointer is smaller than 0, full sign is generated in the Recovered Clock Domain.

The threshold monitor unit counts the valid data in the FIFO and decides to whether produce SKP add or remove request based on its difference with 8.

3.1 SKP deserting operation

As shown in Fig. 5, when the frequency of read clock is slower than write clock, after a certain time, the number of valid date will exceed the threshold of delete, which causes the request for deletion of SKP by threshold monitor. Meanwhile, if the input data is SKP, the SKP delete instruction will effective. As a result, the elastic buffer will suspend the write pointer, then the next data will cover the current data to complete the deletion of SKP symbol. Simultaneously, the data in memory is read out normally, which guarantees the state of elastic buffer at half-full by reducing the number of effective data.

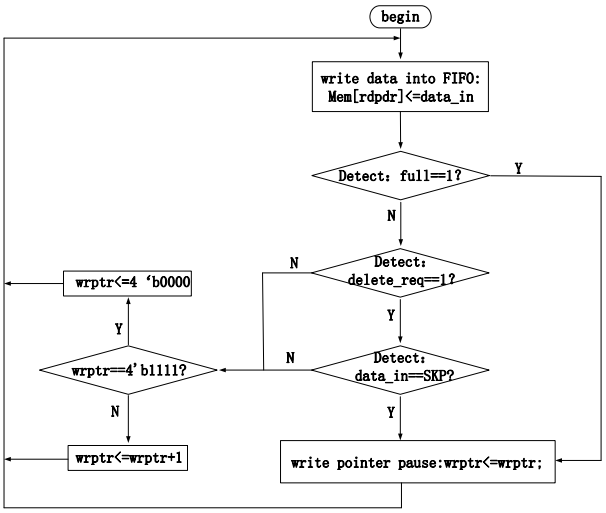

Fig. 5. Flow chart of SKP deleting operation

3.2 SKP inserting operation

When the frequency of read clock is higher than the write clock, after a period of time, the number of valid data is less than the threshold of add in the elastic buffer and then the threshold monitor produces a request signal of SKP adding. Simultaneously, if the output data is SKP symbol, the SKP add instruction will be effective and the read pointer will be paused. The new SKP symbol will be added and outpoint in the next clock cycle. Meanwhile, the new data will be writed into memory normally, which guarantees the state of elastic buffer at half-full by raising the number of effective data, as shown in Fig. 6.

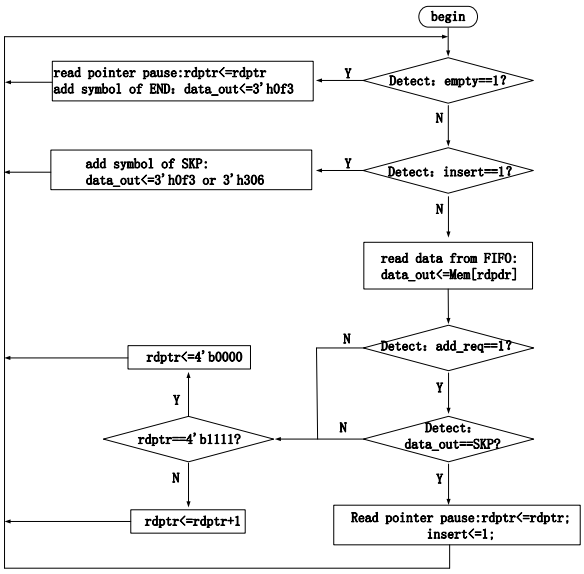

Fig. 6. Flow chart of SKP adding operation

4 Simulation results

In this paper, the proposed elastic buffer is implemented on the FPGA development board with Cyclone IV (Models: EP4CE15F17C8N) and is verified by Agilent logic analyzer (Models: 16803A). According to the literature [4], SKP symbol must be added at least every 1056 data, in order to simulate the output data from CDR, the test signal is inserted a pair of SKP symbols every 1056 data, which is imported into the proposed elastic buffer used in logic verification and testing. Fig. 7 and Fig. 8 are the test results of the proposed elastic buffer, which shows that in the case of inconsistent read and write clock frequency, the proposed elastic buffer can achieve SKP adding and removing function correctly.

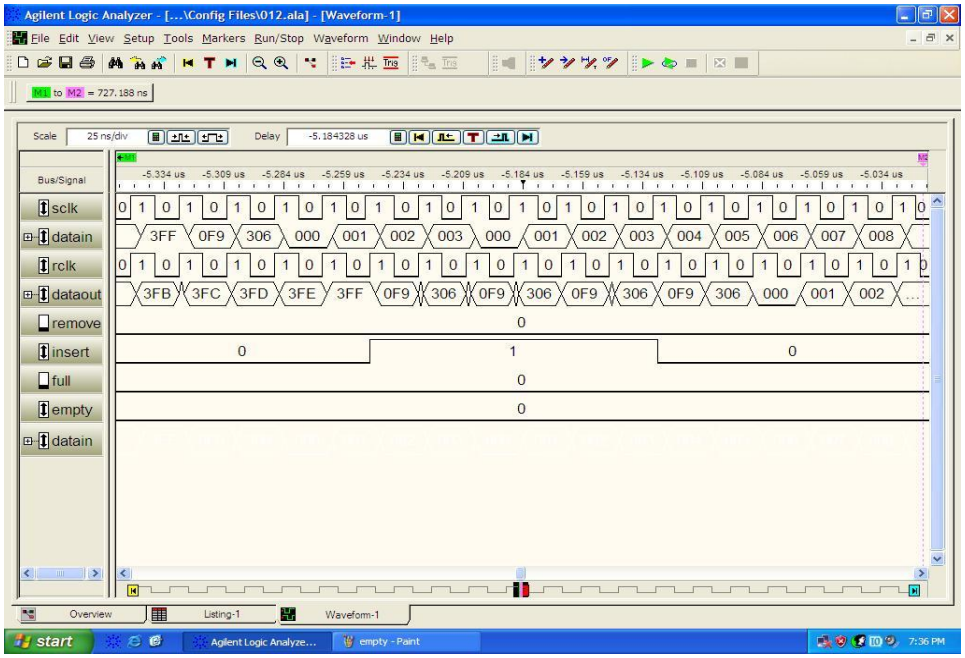

Fig. 7. SKP add process functional diagram

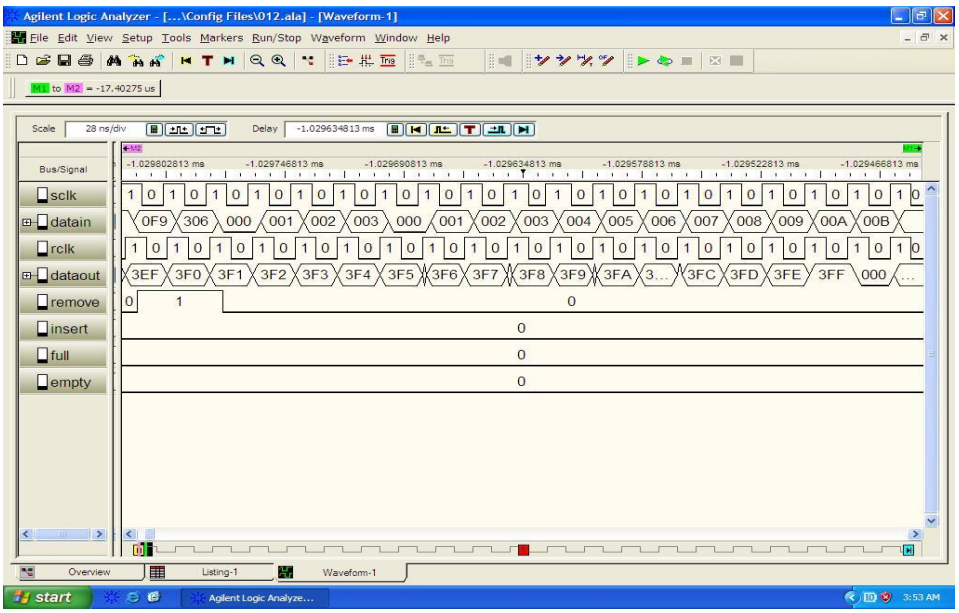

Fig. 8. SKP delete process functional diagram

The proposed elastic buffer has been implemented in TSMC 65nm CMOS technology, and the results of post-simulation are depicted in Fig. 9 and Fig. 10. As can be seen, the frequency offset between two clocks was compensated by adding or deserting SKP symbol at 500MHz.

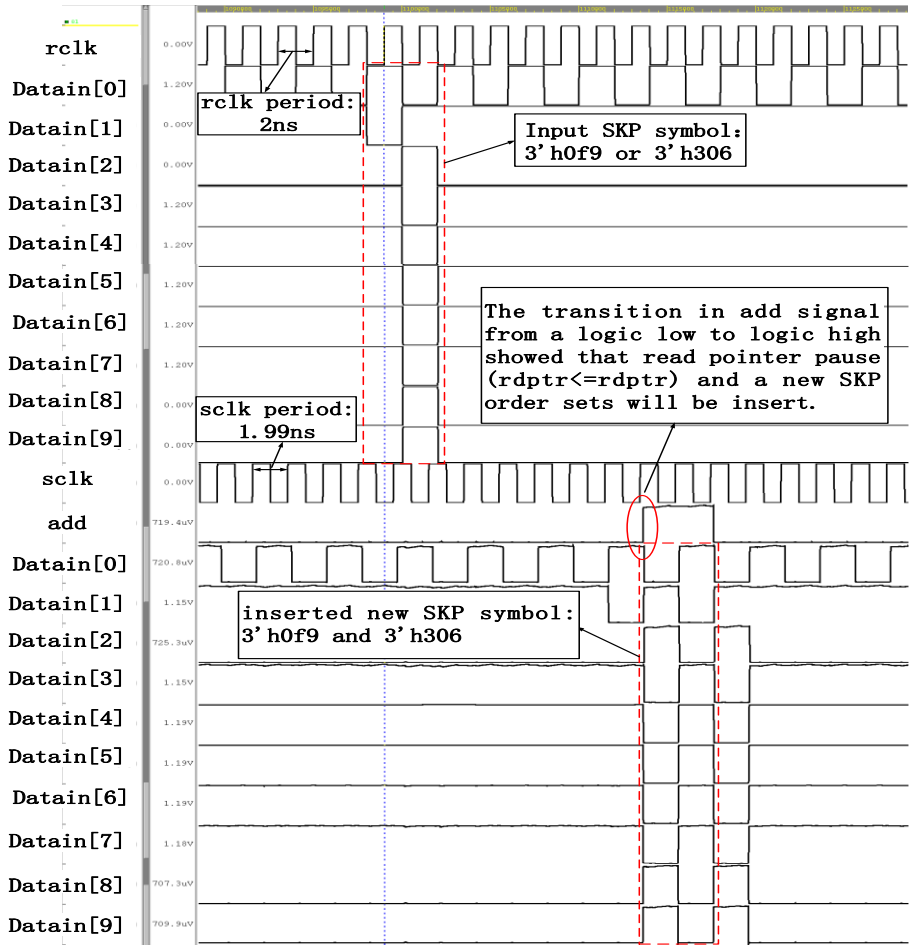

Fig. 9. Post-simulation diagram of timing of add operation

As shown in Fig. 9, the clock cycles of write and read are set to 2ns and 1.99ns, respectively. In this case, the differences between two clocks should be compensated by adding SKP symbol (3'h0f9, 3'h306). When the insert signal is valid, read pointer will be paused and the address of read pointer remains the same, the new SKP symbol will be inserted in the next clock period.

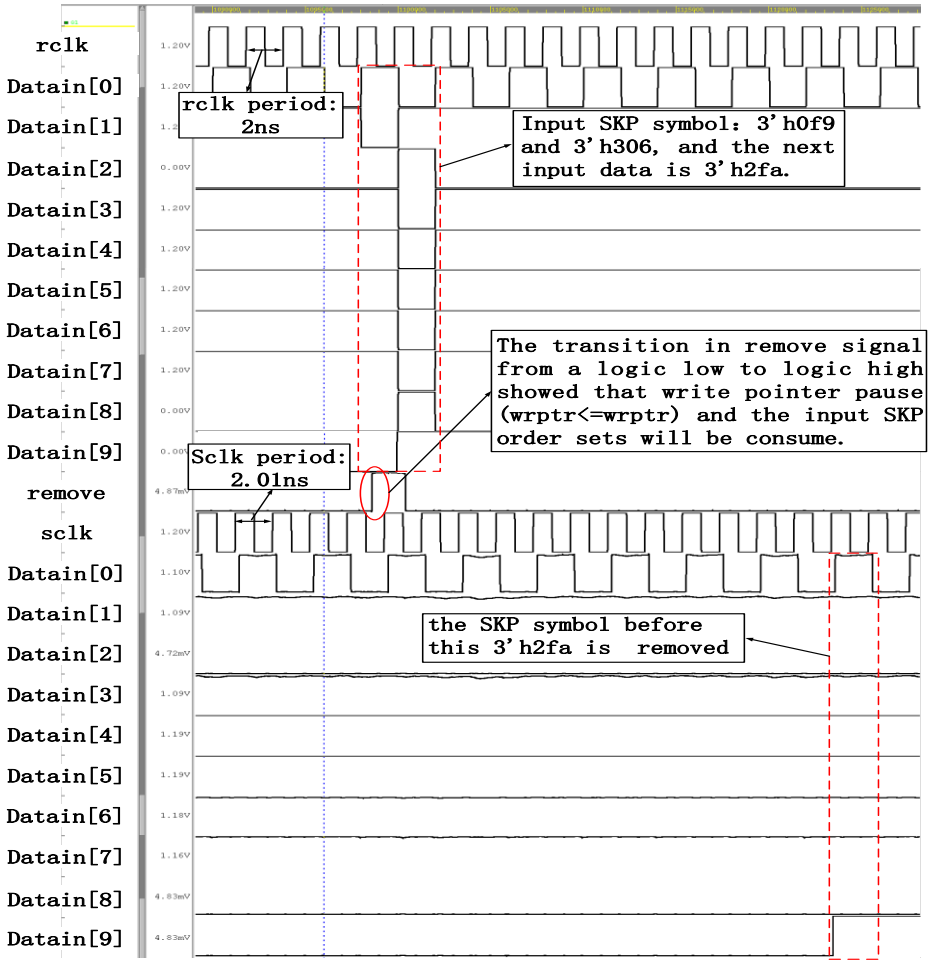

Fig. 10. Post-simulation diagram of timing of delete operation

As shown in Fig. 10, the clock cycles of write and read are set to 2ns and 2.01ns, respectively. The frequency delta between two clocks should be eliminated by SKP deleting process. When the delete signal is effective, write pointer will be paused and the address of write pointer remains the previous state, the SKP symbol inputted before 3'h2fa will be deserted

Table I. Comparison of Performance

Name / performance	Proposed EB	Traditional EB	compare
Power Consumption (mW)	4.164	4.2	↓0.9%
Area (um^2)	11289.6	11315.2	↓0.2%
Read Speed (ns)	0.218	0.220	hold

Table I gives the comparison between the two kinds of elastic buffers. Comparing with the conventional elastic buffer, the proportion of area of this work remains a constant value. In addition, the read speed and the power consumption under the supply voltage of 1.2V also remain unchanged.

5 Conclusions

A new SKP adding technology which is achieved by read pointer suspend is proposed in this paper. Comparing with the elastic buffer proposed in literature 5 and 6, this design avoids the timing errors caused by the asynchronous control between the read and write operations. Additionally, it is no need to write the pointer jumping and breakpoint save, which reduce the use of logical units. The function of the proposed elastic buffer is verified by Agilent logic analyzer. Besides, under the condition of TSMC65nm, the design is synthesized by DC tool and post simulation is accomplished by HSIM. The results of the post simulation indicate that the designed elastic buffer can correctly realize the SKP adding and deleting functions at 500MHz, which meets the requirement of the design of USB 3.0.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61076086 and No.61376098), the Natural Science Foundation for Outstanding Young Talent in Colleges and University of Anhui Province (Grant No. 2012SQRL013ZD)