IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research Paper **© 2012 IJFANS. All Rights Reserved, Journal Volume 11, Iss 01, 2022**

DESIGNING A FIFO BUFFER WITH TWO CLOCKS, FOUR MESOCHRONOUS SLOTS, AND 128 DATA WIDTH

#1VOLADRI PRAVEEN KUMAR, *Asst. Professor,*

#2NARAHARI SUMA, *Asst. Professor,*

Department of Electronics Communication Engineering,

Sree Chaitanya Institute of Technological Sciences, Karimnagar, Ts.

ABSTACT: Unexpected phase connections between clock pulses are possible. Under these situations, clock synchronization is critical for data transfer via modules. We create a novel mesochronous first input-first output (FIFO) dual buffer capable of managing both synchronized and interim storage solutions by autonomously synchronizing data and manually synchronizing only the flow-control signals. The proposed architecture could work. Fully synchronized clocking is being substituted with more adaptive clocking techniques, such as mesochronous clocking, to increase device composability and time elimination. Despite the fact that the transmitter and receiver are separated by a long cable with a delay that does not correspond to the specified operating frequency, the module gets an identical clock signal on both ends of a mesochronous connection and runs at the same clock rate in this configuration. In such cases, the suggested mesochronous FIFO can be built in modular fashion to account for multi-cycle connectivity latencies while retaining the basic principle. Adopting the new architecture is expected to be much less expensive than the current triple mesochronous FIFO systems used by the government. The study also demonstrated the power, latency, and storage efficiency of data widths of 32 and 64 bits.

***KEYWORDS*:**Clock-domain crossing includes source-synchronous communication, mesochronous initially (FIFO), and clock-domain crossover.

1.INTRODUCTION

Modern systems-on-chip (SoC) used in massively scaled technologies are plagued by slow cabling and process (PVT) variances. These issues make asynchronous abstractions across large chip regions more challenging, requiring a significant amount of engineering work to preserve temporal closure. Separating the SoC into globally asynchronous and locally synchronized domains helps to solve the issue because each domain is bound by simultaneous optimization and its timing limitations. The output should be synchronized to the incoming signal domain while crossing clock zones in order to prevent metastability. Along with sending synchronized signals across the time zone interface, it's critical to make sure that any synchronized data that the recipient domains cannot immediately ingest is stored securely until it can be processed. It is essential that these two fundamental and related operations, as well as delay, be linked in a cost-effective way that minimizes latency or storage space overhead because data should be synchronized and (temporary) stored. In this article, we will focus on mesochronous time domains, where clocks run at the same rate but with a continuous, arbitrary phase difference. In these situations, crossing is possible by using a general asynchronous dual first (either first or second) for the mesochronous temporal domain, although this results in an increase in latency. There are currently two approaches to effectively synchronize and buffer over mesochronous interfaces: In a loosely coupled implementation, synchronization and buffering happen separately, while in a closely coupled implementation, they are combined and fused into a single structure. Although the former method can easily handle multi-cycle connection latencies and has the greatest flexibility in terms of system design, it is quite expensive and underutilizes resources. The latter systems, on the other hand, necessitate scrupulous efficiency and little space consumption, but they can only be used with single-cycle timing restrictions or call for phase detectors, which means they could be utilized with multi-cycle delays. This article's overarching objective is to combine the advantages of securely linked and loosely coupled approaches. A unique buffering structure that effectively combines area efficiency, exceptional quality, and support for multi-cycle connection delays is demonstrated in detail. It uses mesochronous synchronization. To our knowledge, the ground-breaking mesochronous dual-clock FIFO architecture is the first to intentionally synchronize only the stream signals in order to achieve data consistency implicitly. This ground-breaking new technology is essential in reducing overall architecture because it eliminates the need to continuously sync the same wide-data buses. In terms of its optimized organization, the suggested synchronizer offers three main advantages: 1) it will be the first layout to achieve lossless operation anywhere at latency depth 1; 2) it imposes minimal buffering requirements for full functionality; and 3) it can be easily expanded to support multi-cycle links without any restrictions. A thorough and demanding hardware test has validated the effectiveness and efficiency of the new design. The suggested solution outperforms the three least relevant current state-of-the-art systems on all key design metrics. Mesochronous Synchronization and Buffering Data can be securely transferred across various mesochronous clock zones using a variety of techniques. The most scalable method uses a "n-flop" mesochronous continuous electric system, as shown in Fig. 1. The then-flop synchronizer in the transmitter domain consists of two free clocks that are monotonically increased every cycle, n parallel registers (n = 4 in Fig. 1), and n parallel clocks. The synchronization message between n flops is stored and retrieved by the two clocks, which are on distinct domains. By employing a reset synchronization structure, the counters are reset to their initial values.

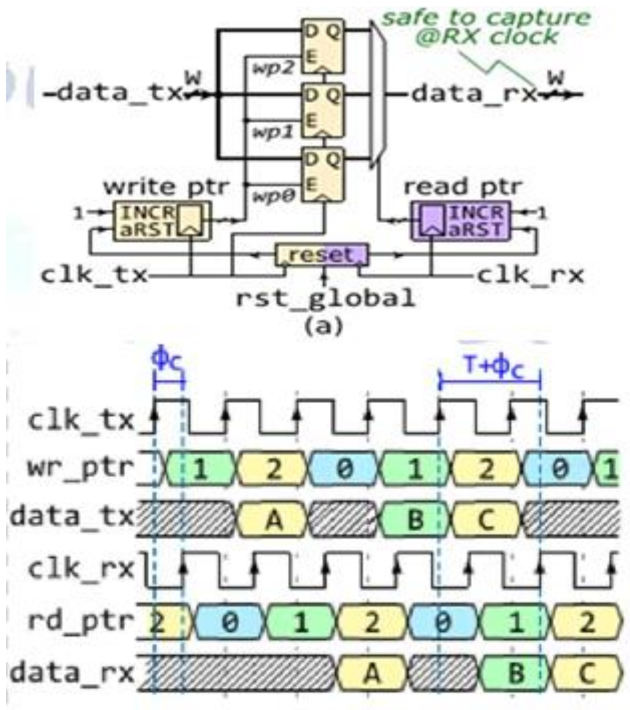


Figure 1: (a) A working example and numerous features for mesochronous clock-domain crossing are both included in a three-flop.

Metastability-free operating is based on the fact that the signal isn't pulled out by the receiver until after a safe interval has occurred following the transmitter writing to that register. By creating a "gap" between the values of the free-running counters, this is accomplished. As explained in the following explanation of Fig. 1, if a register is received by the transmitter clocks at period t, it will be recorded by receiver entries at time t + T + b, where T is the timestamp and c is in fact the TX/RX clock phase skew. (b). n = 2 flips were sufficient in the great majority of phase-skew instances. A 2-flop synchronizer cannot ensure that the output data will remain stable for the receiver clock pulse (i.e., c 0 or c T) when the two clocks are effectively in phase. As a result, a 3-flop relay is required to function securely in any skew period without the usage of a memory cell. A 3-flop relay is sufficient when utilizing a completely deterministic reset mechanism like the one offered. The clocks are turned off during this reset, and they are only turned back on once everyone with access has been reset and all records have gotten their new beginning values. Although the completely predictable behavior of this reset approach tempts, it calls for system-wide clock and reset signal synchronization, which restricts the flexibility of system design. By using an asynchronously reset pulse that is synchronized independently in the transmitter and receiver domains during the startup phase, modern mesochronous synchronizers get around this issue. Dual ruthlessness synchronizer structures are used to synchronize the switching between both domains, as shown in Fig. 2. Depending on the required level of reliability, an additional in-series flip-flop may be incorporated into the brute-force slaver's architecture. In any case, it was shown that, regardless of the phase difference between the two clocks, a 4-flop tangible object is necessary to account for potential quantum superpositions during the initialization stage of (write and perused) countertops and to guarantee that a minimum spread of at least one is achieved after rollback.

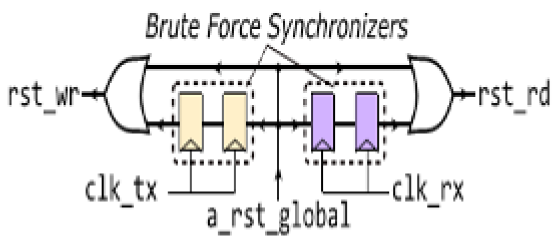
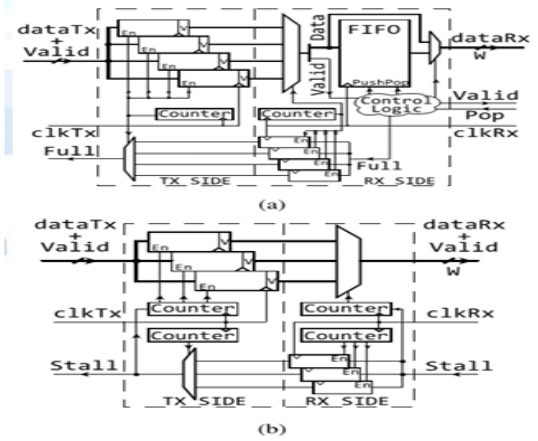


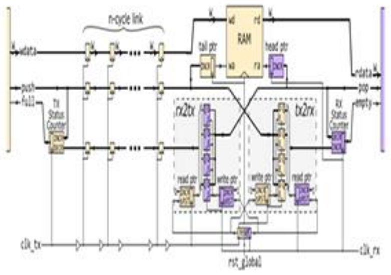
Figure 1 : The counters of the n-flop synchronizer are initialized using an asynchronous reset synchronization structure..

Due to the free feature of the start writing pointer counters in the e-S synchronizer, the receiver is required to read back synchronized data every cycle. It's unlikely that this data will be applied in practice right now. As depicted in Fig. 3, Starsync [8] inserts a synchronized FIFO buffer immediately following data relay to hold synchronized data momentarily. (a). A stretch flow is used on the mesochronous connection to make sure that the transmitter stops sending data when the recipient's caches are full. A forward signal push signal checks the accuracy of the incoming data, and a rearward full signal confirms that the buffer is full. To fulfill the worst-case scenario for synchronized reset initialization, separate single-bit 4-flop slavers are employed to synchronize both signals while they travel across the mesochronous link. There is an alternative organization that offers buffering inside the relay (b), as shown in Fig. 3. In this strongly connected form, no additional buffering is employed; instead, the 3-flop (shown in Fig. 1(a)) serves as a buffer by easing the counter's unrestrained feature. When synchronized data might be processed fast in the receiver due to a downstream stall, the write and read counters are turned off, and the transmitters are instructed to stop providing more data by a stall signal. The tightly coupled organization reduces the space cost because no additional registers are needed to run the FIFO buffer beyond those needed to implement this unique 3-flop synchronizer. Due to this requirement, the transceiver link is subject to extremely strict time constraints, which reduces the potential liaison duration. To overcome the time constraints, a different "hybrid" form of the highly connected organization was created. After reset, the clocks are seen as being in opposition, therefore 3-flop synchronizers work in both cases. If the reset is carried out asynchronously, as in Star sync, a 4-flop real object will be necessary instead. In both loosely linked and densely coupled systems, the size of a lock structure is a characteristic of the selected reset procedure and not an architectural concern. As a result, even though the two systems require different register sizes, they both require the same YE synchronizers because of how they reset.



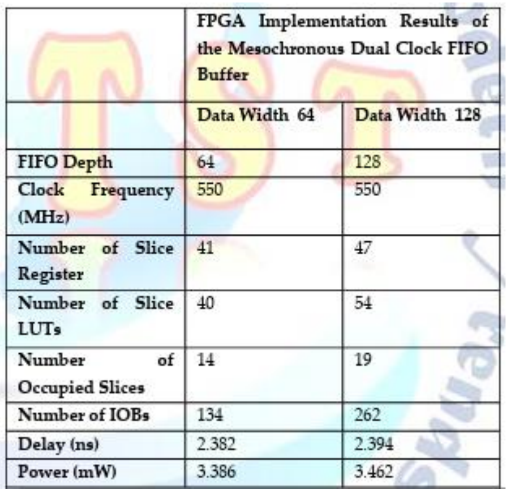
The proposed mesochronous FIFO design avoids the drawbacks of both strongly coupled and loosely coupled systems while combining their benefits. The unique method completely accounts for multi cycle connection delays by combining synchronization and buffering in a low-cost system. The explicit synchronization of the flow-control signals, which is a totally distinct operational approach, synchronizes the data implicitly. Planning and carrying out A Figure 4 depicts the predicted mesochronous FIFO. (a). Data that needs to be synchronized is kept in a buffer in the transmitter domain. Two monotonically increasing counters serve as an index to the memory locations where data is stored and read. Figure 4: Proposed Methodology (a) proposed mesochronous FIFO, which explicitly synchronizes the flow-control stretch signals while also implicitly synchronizing the data. (b) a quick cycle-by-cycle illustration of the operation of the proposed concept. The data is written to the memory address specified by the tail pointer when the transmitter sends a synchronized data word. The pull message to the forward "tx2rx" mesochronous relay is asserted at the same time that the tail pointer is increased [compare Fig. 4(a)].After the demand queuing event is synchronized across the interface, the receiver may safely read data from the memory address provided by the head pointer. This process is demonstrated by the transmission of three data bits ("A," "B," and "C") from Rx to RX in Figure 4(b). After the receiver has completed consuming the food, the state of the queue must be synchronized with the broadcast domain to prevent the queue from overflowing. According to Figure 4, the transmitter keeps its own estimate of the total amount of products that are currently in the queue (a). The counter is increased or decreased whenever an item is directed to perform an action from the queue. To leverage a different reverse improvement that can be achieved since dequeue (pop) events are receiver synchronous, they must be synchronized to the broadcaster's domain. The receivers contest the "rx2tx" synchronizer's push signal during a dequeue.

Depending on the distance between the reading and listening points after reset, there is a forward delay of one to three cycles when synchronizing and enqueuing a piece of data. For safe operation with any phase difference, the answer based on both the reading and listening address at each relay is two. The read pointer is delayed when the push circuit is reset.



The recommended mesochronous FIFO is reorganized whenever the distance between a transmitter and a receiver is too great to fit inside a single cycle. In such cases, the link is separated into numerous register stages for both forward and backward data as well as flow-control signals.

2.COMPARISONS:



	FPGA Implementation Results of the Mesochronous Dual Clock FIFO Buffer	
	Data Width 64	Data Width 128
FIFO Depth	64	128
Clock Frequency (MHz)	550	550
Number of Slice Register	41	47
Number of Slice LUTs	40	54
Number of Occupied Slices	14	19
Number of IOBs	134	262
Delay (ns)	2.382	2.394
Power (mW)	3.386	3.462

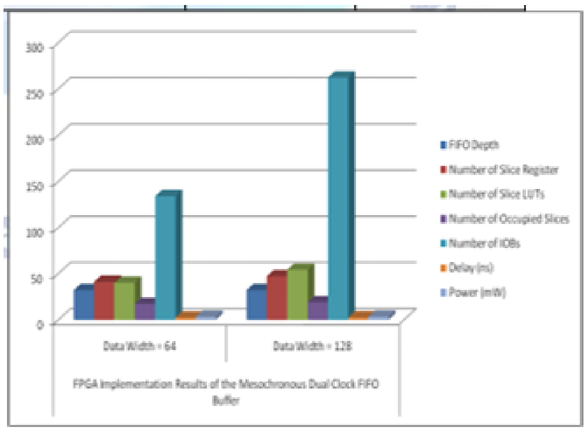


Figure 6: Results of Mesochronous Single Clock FIFO Buffer Implementation.

3.CONCLUSION

In a mesochronous clock interface, the presents a low dual-clock interface. With FIFO, mesochronous clock synchronization and buffering are scalable and independent of the distance between the sender and recipient. The only place where data can be safely sent without being specifically synchronized is on the recipient of a mesochronous connection. For synchronization, only single-bit stretched flow-control signals are used. Particularly in multi-cycle networks, this implicit data synchronization saves a significant amount of space and power without compromising performance.

REFERENCES

1. "Robust interfaces for mixed-timing systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 8, pp. 857–873, August 2004.
2. "Interleaved designs for high-throughput synthesizable synchronisation FIFOs," in Proc. 23rd IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC), May 2017, pp. 41–48.
3. "A review and taxonomy of GALS design styles," IEEE Des. Test Comput., vol. 24, no. 5, pp. 418–428, Sep./Oct. 2007.
4. J. Ax, N. Kucza, M. Vohrmann, T. Jungeblut, M. Porrmann, and U. Rückert, "Comparing synchronous, mesochronous, and asynchronous NoCs for GALS based MPSoCs," IEEE MCSoC, Sep. 2017, pp. 45–51.
5. Digital Systems Engineering, W. J. Dally and J. W. Poulton. Cambridge University Press, Cambridge, U.K., 2008.
6. R. Ginosar, "Metastability and Synchronizers: A Tutorial," IEEE Des. Test Comput., vol. 28, no. 5, pp. 23–35, September/October 2011.
7. R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers across arbitrary and haltable clock domains," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 10, Oct. 2007, pp. 1125–1134.
8. "StarSync: An extended standard-cell mesochronous synchronizer," Integration, vol. 47, no. 2, pp. 250–260, Mar. 2014.
9. D. Verbitsky, R. R. Dobkin, R. Ginosar, and S. Beer, "StarSync: An extendable standard-cell mesochronous synchronizer," Integration, vol. 47, no. 2, pp. 250–260, Mar. 2014.
10. G. N. Gaydadjiev, D. Ludovici, A. Strano, D. Bertozzi, L. Benini, and D. Ludovici, "Comparing tightly and loosely coupled mesochronous synchronizers in a NoC switch architecture," in Proc. NOCS, May 2009, pp. 244–249.