A Four-Stage Mesochronous Synchronizer with Back-Pressure and Buffering for Short and Long Range Communications

DMITRY VERBITSKY, ROSTISLAV (REUVEN) DOBKIN, RAN GINOSAR

VLSI Systems Research Center, EE Dept., Technion, Haifa Israel

A four-stage mesochronous synchronizer is described. It enables low latency and full throughput crossing of boundaries of same frequency, different phase clock domains. Full back pressure is supported, where the receiver can start and stop accepting words without any data loss. Variable depth buffering is provided, supporting a wide range of short and long range communications and accommodating multi-cycle wire delays. Burst data can also be accommodated thanks to buffering. Dynamic phase shifting due to varying voltage and temperature are mitigated by increasing the separation between write and read pointers. Unlike common synchronizers, this circuit is exposed to metastability risk only during reset. The synchronizer provides lower latency and higher reliability than typical two-clock FIFO and other asynchronous and mesochronous synchronizers. It is suitable for implementation using standard cell design and does not require any delay lines, tuning of clock trees and other full custom circuits. It is shown that a minimum of four buffer stages are required, in contrast with previous proposals for three stages.

1. INTRODUCTION

Modern technology generations lead to Systems on Chip (SoCs) comprising multiple IP modules placed on the same die. Clock distribution remains a major issue in such complex systems, because of the wire delay problem and because of delay variations. Distributing the global clock in a system with minimal clock skew is difficult due to the reverse scaling of global wire delay in nanoscale integrated circuits [Friedman 2001].

A fully asynchronous approach to global intra-chip communication would eliminate the clock distribution concerned would make designs more modular since timing assumptions are explicit in the hand-shaking protocols [Sparsø and Furber 2001]. Still, current design tools and IP libraries rely heavily on the synchronous paradigm instead, making intermediate solutions more attractive and affordable in the short run. Generic solutions assuming no knowledge of clock relations may suffer of inferior throughput or may require custom circuits approach [Dobkin and Ginosar 2008; Dobkin and Ginosar 2009]. A trade-off between synchronous and asynchronous approaches consists of the mesochronous scheme [Dally and Poulton 1998; Meng 1990] or the multi-synchronous method [Kol and Ginosar, 1998; Kol and Ginosar, 2000].

In a mesochronous system, a single clock signal is distributed to the various modules in the design with an arbitrary amount of space-dependent time-invariant phase offset (i.e., the skew). Mesochronous synchronization enables architecture scalability and may also mitigate the skew constraints in the clock tree synthesis process, resulting in higher clock rate, lower power and faster back-end turnarounds.

The present paper proposes reliable mesochronous synchronizer architecture with full back-pressure and buffering abilities, for either short or long distance interconnect. Metastability issues are constrained to the reset circuit and are eliminated from the mesochronous synchronizers. The basic synchronizer architecture employs a standard FIFO protocol at its write and read interfaces. Other standard protocols can also be easily employed.

The paper is organized as follows. Sect. 2 discusses related work. Sect. 3 describes the proposed mesochronous synchronizer architecture and design considerations. In Sect. 4 the performance of the proposed synchronizer is analyzed, and the proposed solution is compared with a general two-clock FIFO. Conclusions are drawn in Sect. 5.

2. RELATED WORK

A common approach to the design of mesochronous synchronizers consists of delaying either data or the clock signal to sample data reliably when they are guaranteed to be stable. Figure 1 shows a typical scheme of delay-line based mesochronous synchronizer [Dally and Poulton 1998]. Signal X may change close in time to the sampling clock *clk*, leading to metastability of the XS flip-flop. By changing the delay line settings, the relation between data and the clock is modified so that data transitions happen outside the 'keep-away' region determined by the t_{cy}/t_d delays. A similar approach is suggested in [Kol and Ginosar, 1998; Kol and Ginosar, 2000].

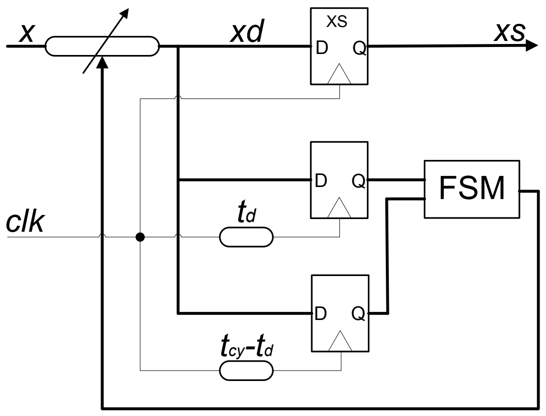


Figure 1 : Delay-line synchronizer [Dally and Poulton 1998]

In [Vangal et al. 2008] a digitally calibrated delay line is employed for shifting the clock of the write-side FSM, while the rest of the transmitter clock domain is clocked by a non-shifted version of the clock. Unfortunately, the principles of choosing the delay value as well as the structure of the synchronizer circuit employed for pointer synchronization were not disclosed. In [Vitulo et al. 2008] a delay line is employed for setting the relative skew between the write and read clocks, and the delay value is determined at design time by static timing analysis (STA). The authors employ double data rate source synchronous communication and sample data at the receiver side using a dual stage buffer. In [Saponara et al. 2008] the method of [Vitullo et al. 2008] is generalized to construct a token-based FIFO-like transfer link.

A different approach for dealing with an unknown clock phase shift involves a phase detector circuits (Figure 2). In [Mu and Svensson 2001] a phase detector is used to predict conflicts and to delay the write clock. Similar methods of phase detection for conflict prediction are proposed in [Mesgarzadeh et al. 2002] and [Wiklund 2003], where for each predicted conflict the write clock is shifted as well. Kol and Ginosar [2000] employs conflict detectors to track drifting phase shifts between multi-synchronous clock domains. In [Semiat and Ginosar 2003; Frank and Ginosar 2004] conflict detectors are employed to prevent metastability when using periodic clocks.

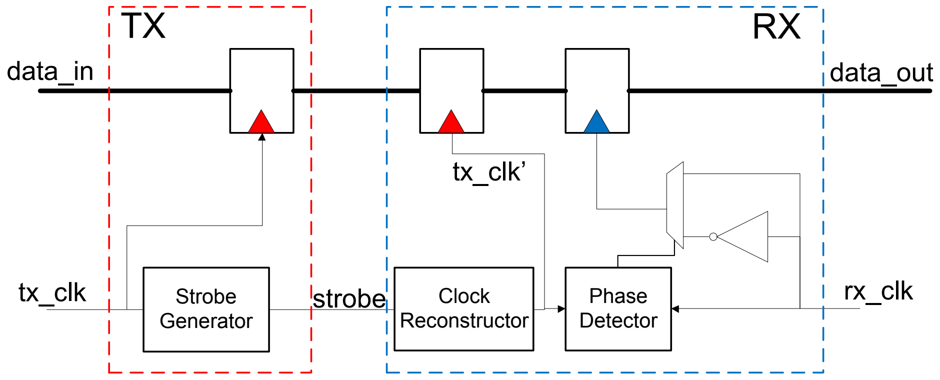


Figure 2 : Phase Detection Synchronizer

Long wire delays between communicating modules typically incur additional skew. In [Caputa and Svensson 2006] an N-depth mesochronous synchronizer is proposed to deal with global wire delays, estimated using STA and used to define the spread of the write and read pointers of the N-depth synchronizer. In [Ghoneima et al. 2007] a similar technique for long wires is proposed, but the pointer spread is calculated dynamically, using a phase detector.

Delay-line based synchronizers are mostly suitable for full custom designs. They require clock tree modifications and multiple instantiations for multi-bit applications. For SoCs which are based on standard cell design, including both ASIC and FPGAs, and which do not permit any fine-tune clock tree modifications, another approach to mesochronous synchronization is required. The principle structure of such a synchronizer is shown in Figure 3. This mesochronous synchronizer employs cyclic write and read pointers with a certain initial spread to allow collision-free write and read operations. Jex and Dike [1995], as well as Alshaikh et. al [2010], employ a similar structure; the goal is to provide time for multi-stage metastability resolution, avoiding a serial synchronization chain.

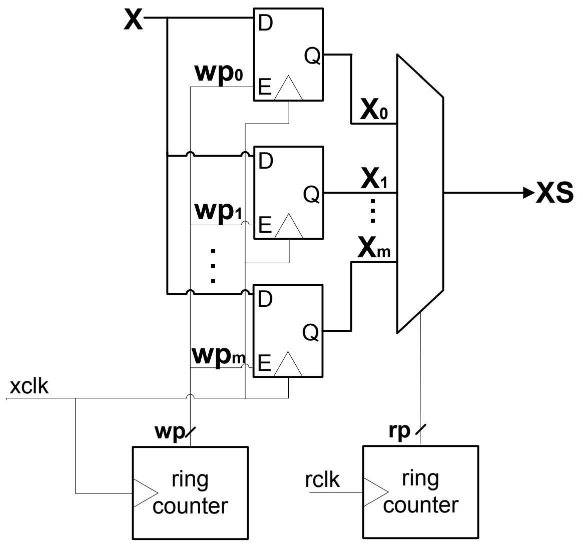


Figure 3 : *m*-elements cyclic buffer mesochronous synchronizer [Dally and Poulton 1998]

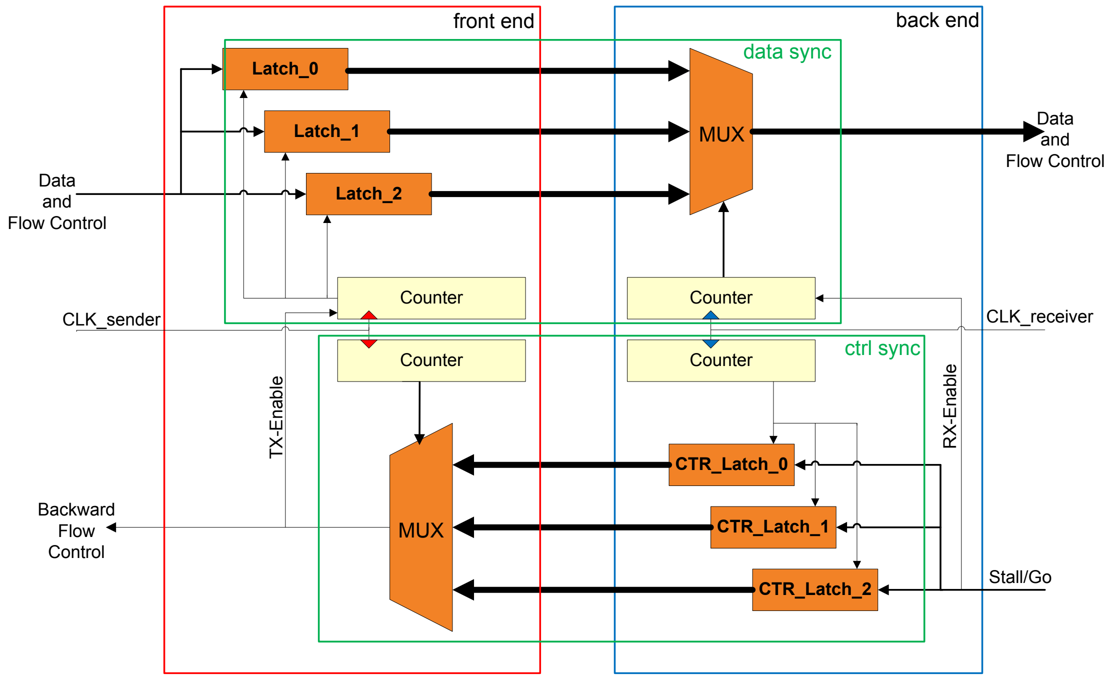


Figure 4 : A Three-Element FIFO Synchronizer with Back-Pressure [Ludovici 2009]

Ludovici et al. [2009] employ a three-element cyclic buffer mesochronous synchronizer as shown in Figure 4. The architecture comprises two synchronization paths. The 'data sync' path synchronizes data that are sent from the transmitter to the receiver. The 'ctrl sync' path synchronizes back-pressure signaling from the receiver to the transmitter. Both synchronization paths employ cyclic buffers, each consisting of three stages. Buffers are managed by cyclic counters, and their initial spread should avoid metastability. Counters of the ‘ctrl sync’ run continuously, whereas counters of the ‘data sync’ are stopped each time the Stall signal is received (TX/RX Enable in Figure 4).

This paper proposes a new architecture for data transfer between mesochronous clock domains. In the proposed architecture the receiver clock is never stopped or modified, and the write and read pointers are never stopped as well. The architecture supports full backpressure and buffering for data burst transfers and is also suitable for long range communications. In addition, relative to other implementations, in the proposed architecture only a single cyclic-counter is employed at each side of the synchronizer. The initial spread of the pointers and the number of stages in the synchronizer are the most important design considerations. The paper analyzes pointer spread and the number of stages required for different operating conditions in Sect.3.2. We show that three synchronization stages are insufficient and may lead to metastability; rather, a minimum of four stages is required.

3. FOUR-STAGE MESOCHRONOUS SYNCHRONIZER WITH BACK_PRESSURE AND BUFFERING

The proposed architecture is described in Sect. 3.1, and synchronizer initialization is discussed in Sect. 3.2.

3.1. Architecture

The proposed architecture is shown in Figure 5. The four-stage mesochronous synchronizer employs two cyclic buffers (TX_BUF and RX_BUF), each consisting of *four* parallel stages. TX_BUF is a data buffer, responsible for data transfer from the transmitter interface to the receiver. TX_BUF holds both forward tokens and data. Forward tokens indicate the data validity. RX_BUF holds backward tokens, which indicate the amount of free space at the receiver side.

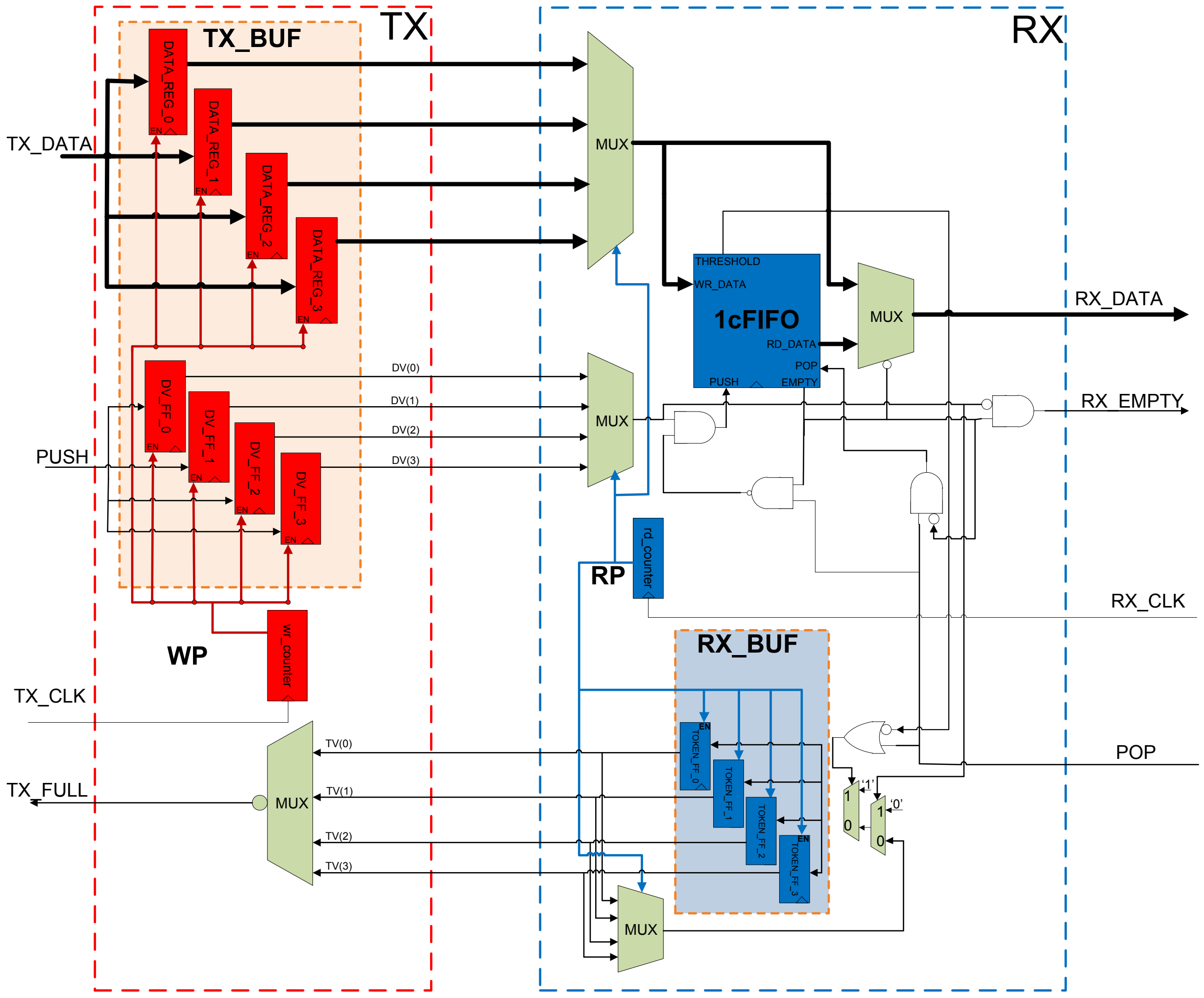


Figure 5 : Mesochronous Synchronizer structure

Data Tranfer. The data and the forward token (PUSH signal converted into data valid DV signal) are written cyclically into DATA_REG_x and DV_FF_x registers with WR_CLK. The cyclic WR_COUNTER is never stopped. Same is true for the RD_COUNTER, which is clocked by the mesochronous RD_CLK. Thanks to the initial spread (discussed in Sect. 3.2) metastability is avoided when reading data from TX_BUF using the read pointer RP. The read data are output over the RD_DATA and EMPTY lines. The EMPTY signal is an inverted copy of the DV signal.

Backpressure. The receiver side can stop the transfer by de-asserting the POP signal, which immediately directs the data from TX_BUF into the Single-Clock FIFO (1cFIFO) instead of the data output RD_DATA lines. Note that if there are valid data in TX_BUF or in 1cFIFO then the EMPTY signal remains low, since it always indicates the presence of valid data regardless of whether the data are held in TX_BUF or in 1cFIFO.

The backward token in Token_FF_x is either 'ClearToSend' ('1'), allowing writing into DATA_REG_x and DV_FF_x, or 'DoNotSend' ('0'). In a basic (minimal buffering) configuration, the 1cFIFO depth is equal to the depth of TX_BUF (four buffers), to allow clearing of TX_BUF into 1cFIFO, since the pointer counter is never stopped. 'DoNotSend' tokens are written into RX_ BUF when POP is de-asserted. When Token_FF_x is 'DoNotSend', writing data into the corresponding DATA_REG_x and DV_FF_x is blocked, since the data register is occupied. TX_FULL signal is asserted, providing back-pressure. Data transfer is resumed when POP signal is re-asserted. After asserting POP, data from 1cFIFO are output on RD_DATA. In parallel, BUF2 is filled with 'ClearToSend' tokens that are passed to the transmitter. Once a 'ClearToSend' token is selected by WP, TX_FULL is de-asserted, enabling a write operation. During subsequent write and read operations, the data at the receiver side are pushed into 1cFIFO as long as the FIFO is not empty. Once the FIFO becomes empty (e.g. thanks to idle cycles at the transmitter side) the data bypass the 1cFIFO and are output with minimal latency.

Burst support and buffering. As described above, the minimal 1cFIFO depth equals the depth of TX_BUF (depth of four in the basic configuration). The minimal depth is dictated by the maximal number of words inside TX_BUF. Since the read out of TX_BUF is never stopped, the words are saved aside in the 1cFIFO when POP is de-asserted.

Burst transfer is supported by making 1cFIFO deeper. For a data burst of length L>4 the 1cFIFO depths should be set to be L. The THRESHOLD signal is asserted when the number of data words in 1cFIFO reaches a predefined threshold of L-4. As described below, both POP and 1cFIFO's THRESHOLD affect the tokens generation for RX_BUF.

Token Management. Tokens buffer RX_BUF is managed in the following way:

- All token buffer stages are initialized to '1' (Valid Token)
- For valid read cycle (namely, POP is high) the following operations are performed:
  - Valid Token ('1') is written into the current RX_BUF stage (pointed to by read pointer RP), indicating that the corresponding TX_BUF stage is empty and is ready to receive new data.
  - When 1cFIFO is not empty, current data from TX_BUF is written into 1cFIFO (only if the current TX_BUF register holds valid data, namely the corresponding DV is high) and the data for RD_DATA bus is popped out of the 1cFIFO. Otherwise (when 1cFIFO is empty) valid data from TX_BUF are conveyed directly to the output (1cFIFO is bypassed).
- For stall cycle (namely, POP is low) the following operations are performed:
  - Valid data from TX_BUF are pushed into 1cFIFO.
  - When THRESHOLD is low (namely, there is still room in the 1cFIFO), a Valid token is written into RX_BUF.
  - When THRESHOLD is high (namely, the 1cFIFO has room only for words that are already in TX_BUF but no room for additional new words), Invalid token ('0') is written to RX_BUF. Note that 1cFIFO has room also for new words that may be

inserted into TX_BUF until the Invalid backward token has propagated and toggled the TX_FULL signal.

Since WR_CLK and RD_CLK have the same frequency, but a constant relative skew, the initial values of the write and read pointers should be carefully set to eliminate contention (WAR or RAW hazards). The initial write pointer is set to 0, and the initial read pointer is set to 2 (see Sect. 3.2 for details). The initial separation is preserved throughout subsequent operation thanks to the mesochronous clock relationship.

At the output, the RD_DATA multiplexer chooses the output of 1cFIFO, when it is not empty, or TX_BUF otherwise. Additional multiplexing selects RD_EMPTY, which is '0' if either internal 1cFIFO is not empty or internal 1cFIFO is empty but the TX_BUF holds valid data, and '1' otherwise.

Long Wires. Here we consider two different cases of communication over long-interconnect. The interconnect is considered long when the wire delay is significant, e.g., longer than one clock cycle:

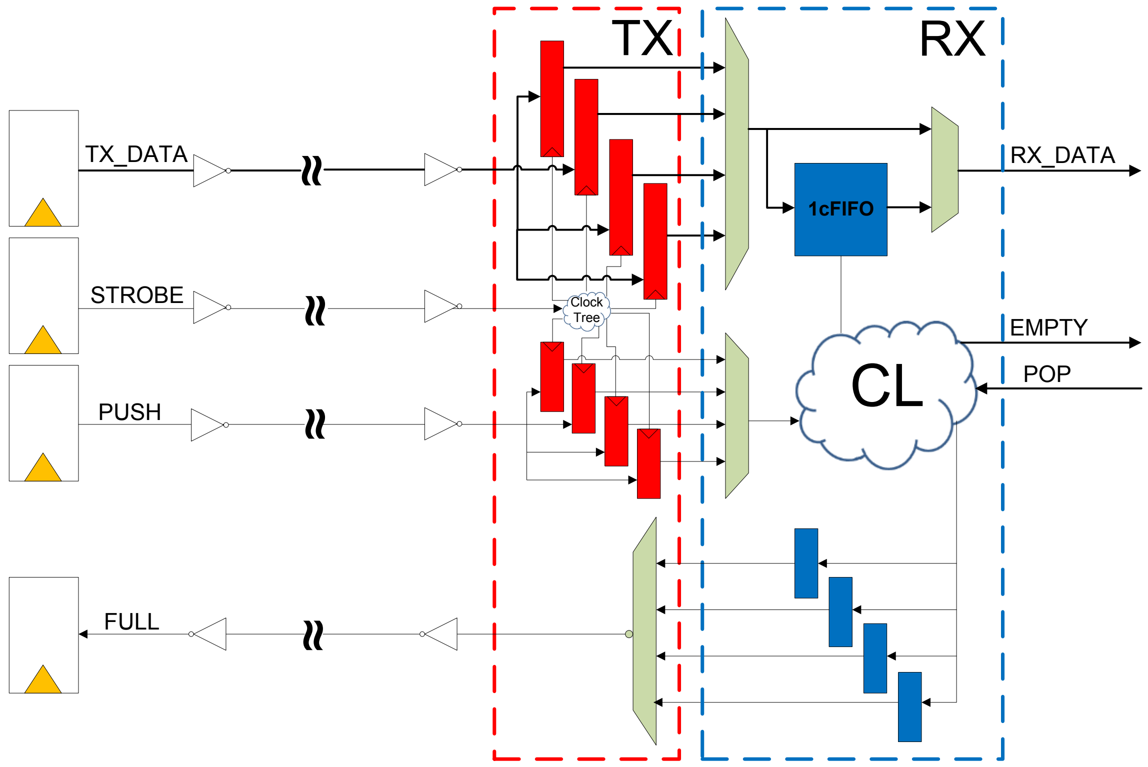
1. Source synchronous communication, where the entire synchronizer is placed at the sink of the communication link (Figure 6a). A similar approach was presented in [Ghoneima et al. 2009].
2. Split synchronizer: Long wires between the TX and RX sides of the synchronizer (Figure 6b).

The proposed four-stage synchronizer (Figure 5) requires no modifications for the former (source synchronous communication) case, when no backpressure is applied (POP is always asserted when EMPTY is low). When the receiver RX is stopped, the TX_FULL indication may arrive in the link source module (Figure 6a) after a few clock cycles due to the long interconnect delay. By increasing the depth of the 1cFIFO, the proposed architecture easily deals with this case without any latency overhead. The additional required buffer size can be computed using STA. Note that any spare stages added for safety affect only the area; neither throughput nor latency are affected.

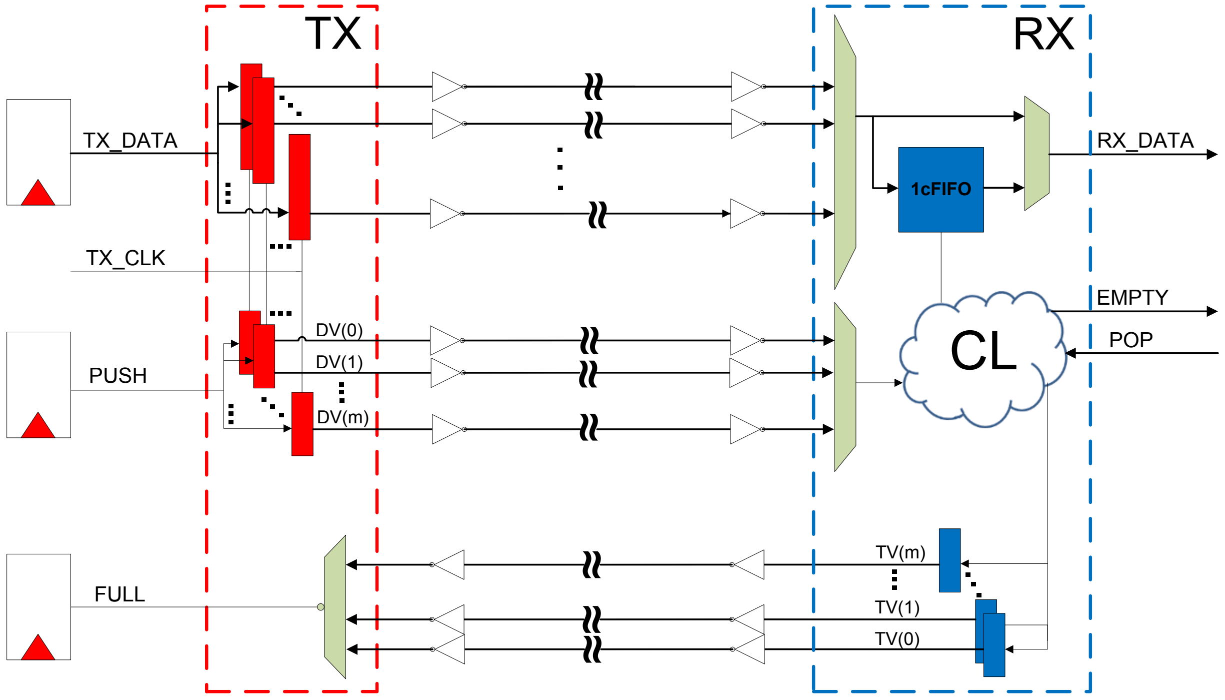
In the split synchronizer case, in which the TX and RX parts are placed far away from each other, long delay is incurred both over the TX to RX data path and over the RX to TX token path. When no backpressure is employed, only the TX to RX interconnect delay is taken into account. In this case, the synchronizer should assure that the data are read before the same buffer is written. Since it takes time for the data and for the DV signal to reach RX, the synchronizer TX_BUF depth is enlarged (by configuration) to cover for the longer interconnect delay (Figure 7a). The additional number of stages is computed using STA. When backpressure is desired, additional latency of the backpressure signal TV (Token Valid) should be taken into account (Figure 7b). This is similar to the source synchronous case: the depth of 1cFIFO is reconfigured to compensate for that delay. Note that if the backward and forward flight times are similar, k=m in Figure 7b. This split TX/RX configuration with long-wire interconnect is useful for mesochronous NoC, where TX can serve as the output port of a NoC router and RX is the input port of the next router.

Multi-synchronous communications. Mesochronous clocks may suffer of phase drifting during operation, due to temporal and spatial variations in voltage and temperature. In this work we analyze dynamic phase shift in the range of $[0, \pm T)$ between the clocks. We show in the next section that six synchronization stages can deal with such phase drifts.

In general, each additional phase shift of T adds two synchronization stages. It means that in order to cope with phase shift of $[0, \pm kT)$ we need $4 + 2k$ stages in each of TX_BUF and RX_BUF.

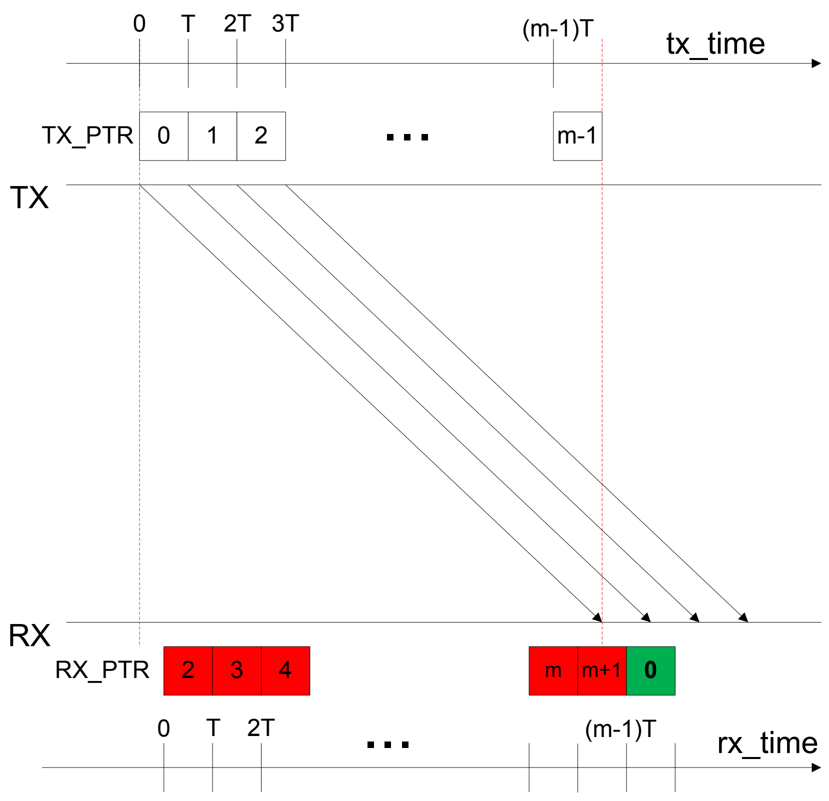


(a)

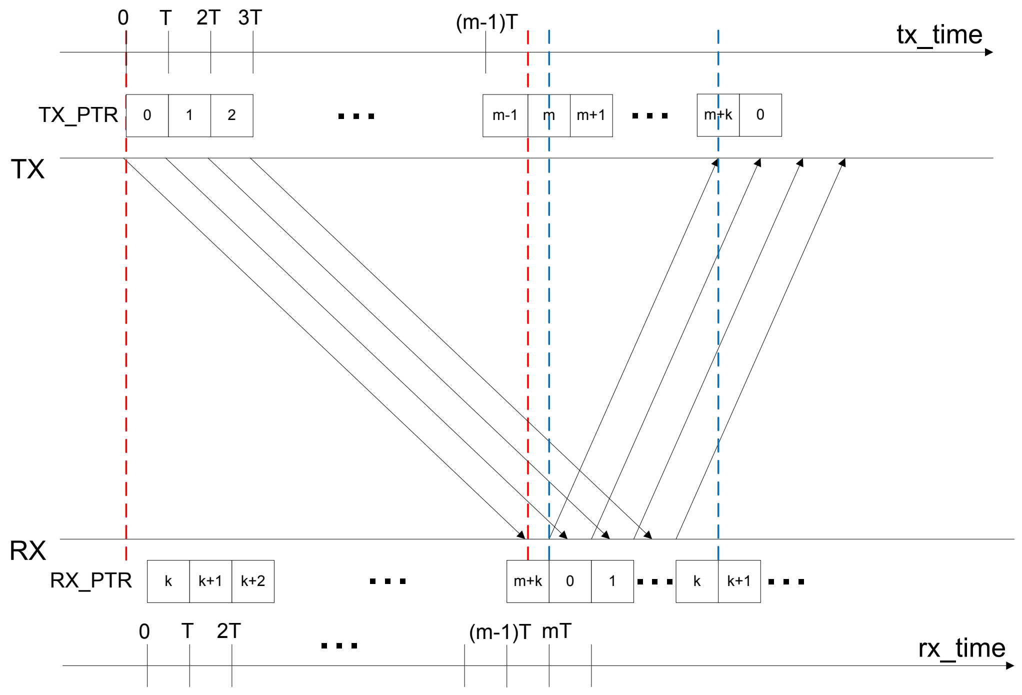


(b)

Figure 6 : Communication over long interconnect (a) Source synchronous communication (b) Split synchronizer: Long wires between TX and RX (DV and TV signals)



(a)



(b)

Figure 7 : data/token transitions (a) no backpressure (b) with backpressure

3.2. Initialization issues

An asynchronous reset is employed for circuit initialization. Note that the non-determinism discussed below is present in any system since no system can provide synchronous reset to both read and write clock domains without synchronization. A straightforward synchronization employs a two flip-flop synchronizer (we assume here that two flip-flops are sufficient in terms of MTBF [Ginosar 2005]). More generally, an external asynchronous reset is synchronized to both read and write clock domains by means of N flip-flop synchronizer, as shown in Figure 8 (N=2 is shown).

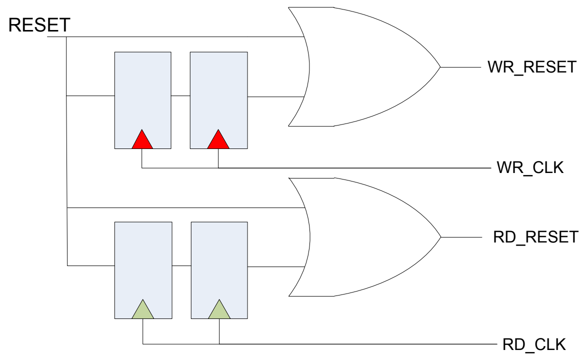
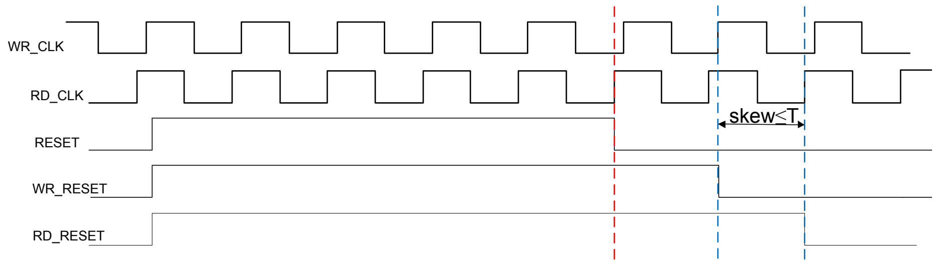
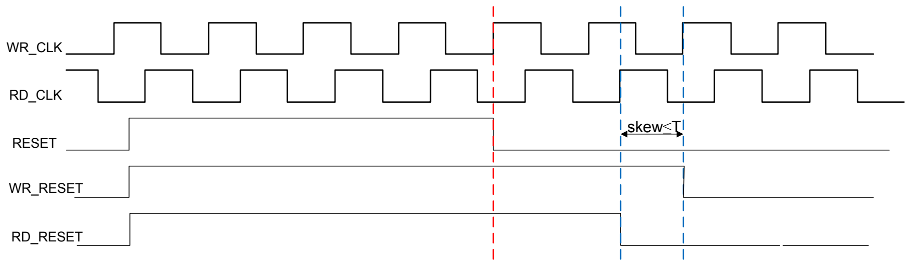


Figure 8 : Synchronization of asynchronous RESET

For each clock domain the reset is asserted asynchronously and instantanesously, but is released synchronously, leading to skew of [0, ±T), where T is the clock period. The following waveforms describe the operation for two possible maximum skew scenarios that can occur.



(a) WR_RESET precedes RD_RESET



(b) RD_RESET precedes WR_RESET

Figure 9 : Reset: two maximum skew scenarios (a) positive skew up to T (b) negative skew up to T

When resets are asserted, write and read pointers are set to their initial values (0 and 2 respectively, for short interconnect, or 0 and *k* for long interconnect with backpressure). These initial values are explained as follows. To prevent metastability, the circuit must assure that the write and read pointers are set correctly. The first requirement is that the initial pointer values cannot be equal. The situation is described in Figure 10. It is shown that if both WR_PTR and RD_PTR are set to the same initial value, there is a hold-time violation. Thus, read and write pointers must be spread by at least one place.

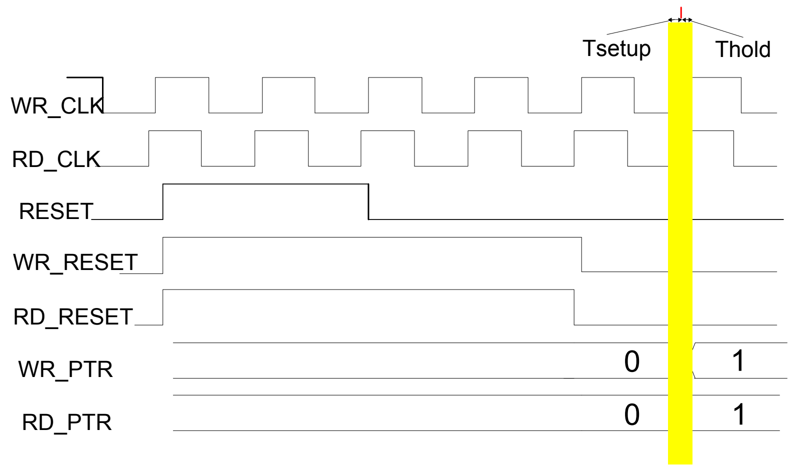


Figure 10 : Hold time violation when pointers are initialized to the same value

Figure 11 shows that the sufficient pointer spread equals two since for spread of one, hold time violation is still possible. One can see that when data are read from TX_BUF with RD_PTR=1 there is a concurrent write to the same TX_BUF stage (WR_PTR = RD_PTR = 1). Thus, the most optimal solution would be ***four register stages*** with initial write pointer=0 and initial read pointer=2.

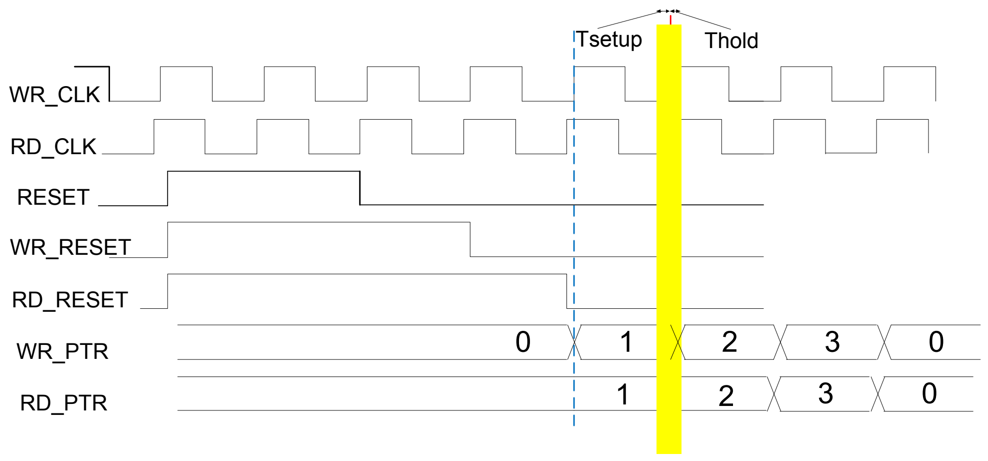


Figure 11 : Hold time violation when pointers are initialized with spread of one

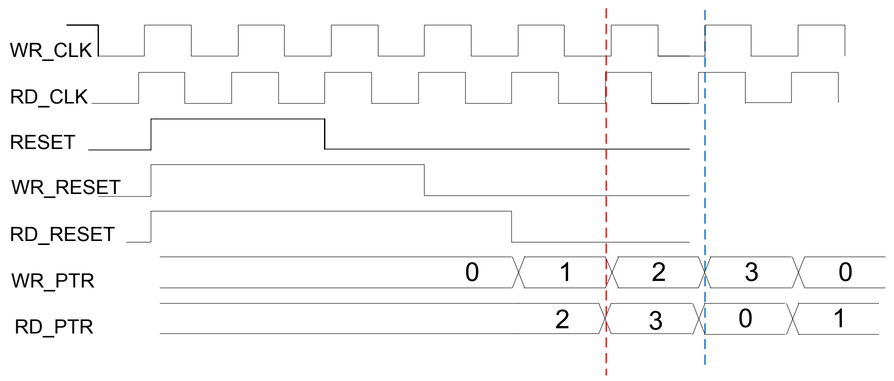


Figure 12 : Reliable synchronization

The synchronizer can be enhanced to support a dynamic phase drift. Here we analyze a phase drift limited to a single clock cycle. After a phase drift of a whole cycle the spread between the pointers is reduced and the situation described in Figure 11 is possible. In order to avoid this, additional spread of one place must be added between the pointers. That means that the initialization setup would be six register stages with initial write pointer = 0 and initial read pointer = 3. In Figure 12 it is shown that read and write from the same index never happen in the metastability regions of neither clock domain.

4. PERFORMANCE

In this section the performance of the four-stage mesochronous synchronizer is analyzed. Forward latency, throughput, area and power requirements are discussed. The proposed solution is compared to a two-clock FIFO.

4.1. Forward Latency

The forward latency is defined as the time from the data being placed at the synchronizer input until it appears at the synchronizer output. The forward latency varies in the range (T, 3T). The latency depends on the initial pointer spread and on the relative phase difference of the two clocks. Figure 13 shows the forward data latency when RD_CLK has a positive phase shift. Figure 14 shows a forward data latency when RD_CLK has a negative phase shift. Φ is the minimal absolute phase shift value between the clocks.

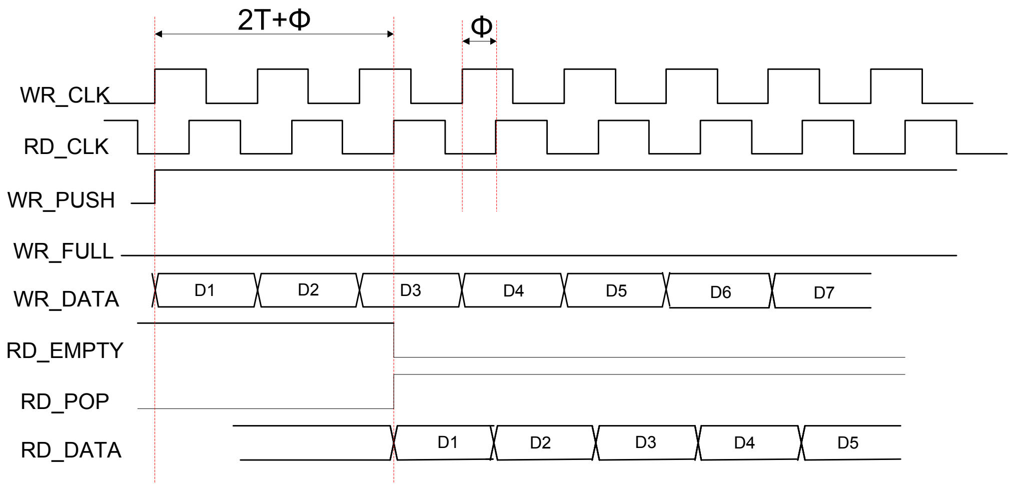


Figure 13 : write reset is released before read reset

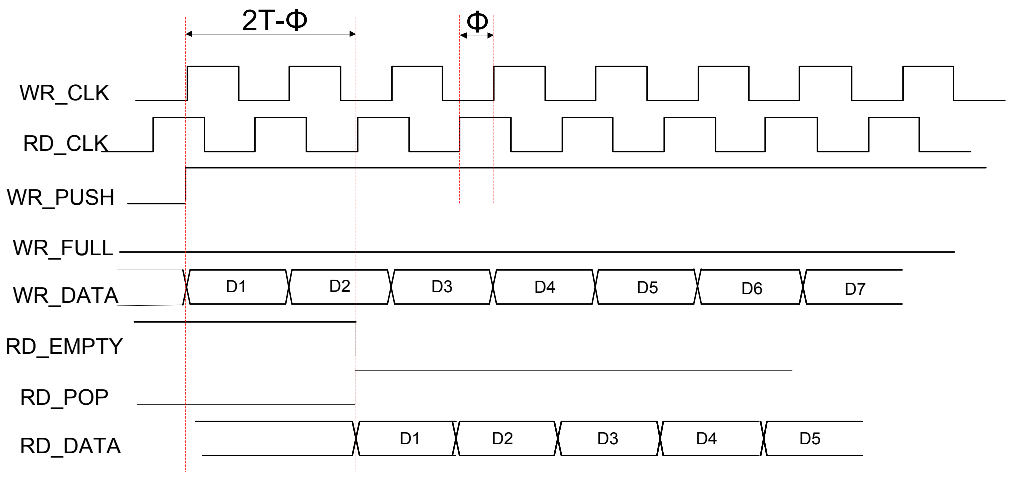


Figure 14 : write reset is released after read reset

4.2. Throughput

The four-stage mesochronous synchronizer provides maximal throughput, namely a data word can be transferred on each clock cycle, when the receiver side is ready and if there is no backpressure. When backpressure is applied, the throughput is linearly affected as shown in Figure 15.

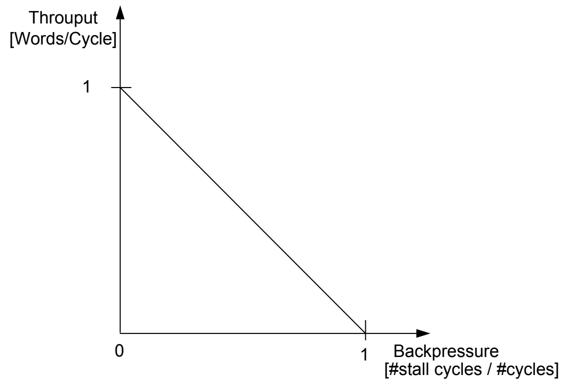


Figure 15 : Throughput vs. RX backpressure

4.3. Area

The number of registers in the four-stage mesochronous synchronizer depends on the word data width (*DataWidth*), on the depth of the synchronization buffers (*BufDepth*), and on the burst length (*BurstLength*) as shown in Eq. (1). The register data count comes from two main sources: one is the TX and RD BUFs and their pointers WP and RP, and the 1cFIFO. Note that *BufDepth* and *BurstLength* should be greater or equal to four. Increasing *BufDepth* (in the case of long range interconnect) has the most significant impact among all the parameters, as it affects also the minimal value of *BurstLength*.

$$\begin{aligned}
&Area = \\
&BufDepth \cdot (DataWidth + 2) + \quad \#\text{TX\&RX BUFs} \\
&2 \cdot \lceil \log_2(BufDepth) \rceil + \qquad\quad \#\text{WP \& RP} \\
&BurstLength \cdot DataWidth + \lceil \log_2(BurstLength) \rceil \quad \#\text{1cFIFO}
\end{aligned} \tag{1}$$

4.4. Dynamic Power

In the four-stage mesochronous synchronizer a single register inside each buffer is enabled per cycle thanks to the cyclic enable pointers. Thus, data toggles in only about one fourth of the circuit. In addition, 1cFIFO works only when backpressure is applied, while during continuous data transfer it can be clock-gated. Therefore, for a continuous data transfer, the power is proportional to about one fourth of the TX/RX_BUF registers. Clock gating can be employed in all the disabled registers, controlled by WP and RP. Thanks to the mesochronous clock relation (and the fact that typically the two clock phases are different from each other), peak power is reduced since TX and RX clocks do not toggle simultaneously.

$$\begin{aligned}
&Power \propto \frac{1}{4} \times \\
&\begin{bmatrix} BufDepth \cdot (DataWidth + 2) + \\ 2 \cdot \lceil \log_2(BufDepth) \rceil \end{bmatrix}
\end{aligned} \tag{2}$$

4.5. Comparison to 2-Clock FIFO

A general structure of two-clock FIFO is shown in Figure 16. Incoming data are pushed into the FIFO when write enable is asserted (push). On the subsequent cycle, a gray coded write pointer wp_g is sent to synchronizer, which usually consists of at least two stages. The synchronized value of the write pointer, wp_sync, is then compared to the current value of the read pointer to assess the amount of unread data inside the FIFO, setting the empty signal accordingly. When the empty is de-asserted, pop signal can be applied (for fast response the pop signal can be asserted immediately after empty de-assertion, on the same clock cycle). In a standard FIFO protocol, the data appear at the FIFO output on the next clock cycle.. For fall-through FIFO the data appear along with the empty de-assertion.

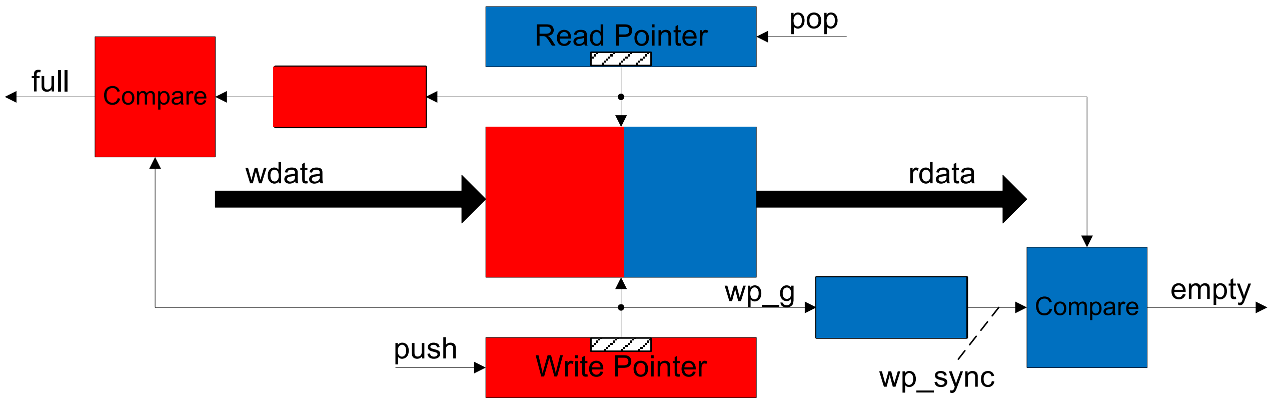


Figure 16 : 2-clock gray fifo structure[Cummings 2002]

Figure 17 shows the waveform diagram of the two-clock FIFO operation and the forward latency for the standard FIFO protocol. The total worst case latency is four clock cycles. For the fall-through FIFO it can be reduced by one cycle. We assume that both Empty and POP are not registered, otherwise an additional two clock cycle latency should be added to the forward latency. In terms of throughput, both structures provide the maximal throughput of one word per cycle (WPC). The area expression for the two-clock FIFO is shown in Eq.(3) We assume here that the two synchronization stages are enough to obtain high reliability, otherwise the factor of 4 in Eq.(3) should be increased.

$$\begin{aligned} &Area = \\ &2\cdot\lceil \log_2(BurstLength)\rceil + \quad \#\text{WP \& RP} \\ &4\cdot\lceil \log_2(BurstLength)\rceil + \quad \#\text{SYNCs} \\ &BurstLength\cdot DataWidth \quad \#\text{Buffers} \end{aligned} \qquad (3)$$

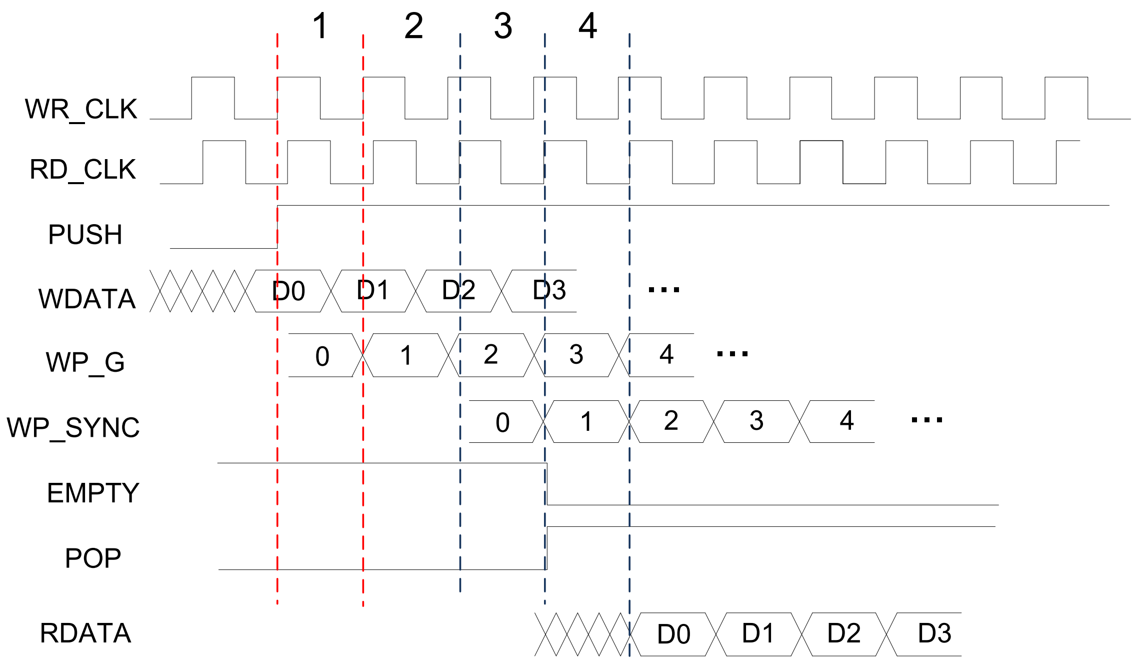


Figure 17 : Two-Clock FIFO latency

Table 1: Comparison of the four-stage mesochronous synchronizer to two-clock FIFO

Parameter	Four-stage mesochronous synchronizer (This paper)	Two-Clock FIFO solution (with two flip-flop synchronizer)
Latency (cycles)	3	3—6
Throughput (WPC)	1	1
MTBF	Infinity	Technology & Operating conditions dependent

The four-stage mesochronous synchronizer is compared to the two-clock FIFO in Table 1. The latency of the mesochronous synchronizer is at most the same as the minimal latency of the two-clock FIFO, when the same maximal possible throughput is provided. The latency of the two-clock FIFO depends on the clock rate and the technology as for certain cases a longer delay is required to resolve metastability inside SYNC modules. The mesochronous synchronizer, however, does not incur such overhead and will always exhibit three or fewer cycles forward latency (down to one cycle), with an infinite MTBF. In addition, the two-clock FIFO does not support long wire communication (Sect. 3.1) since it cannot be distributed.

To provide full-throughput, the two-clock FIFO is usually designed with a depth of at least eight places. Figure 18 compares the area requirements for different data word sizes for the mesochronous (M-Sync) and two-clock FIFO (2cFIFO) with minimal buffering that allows full throughput of 1 WPC. As can be observed, the M-Sync requires less area than the two-clock FIFO. When the data width is held constant, and the burst length is high, both Eq. (1) and Eq. (3) converge to the value of $BurstLength \times DataWidth$, thus both architectures require approximately the same area when a long burst is required.

In terms of power, we note that the two-clock FIFO must always work with its internal dual-port RAM, whereas the mesochronous synchronizer may bypass the RAM consuming less dynamic power.

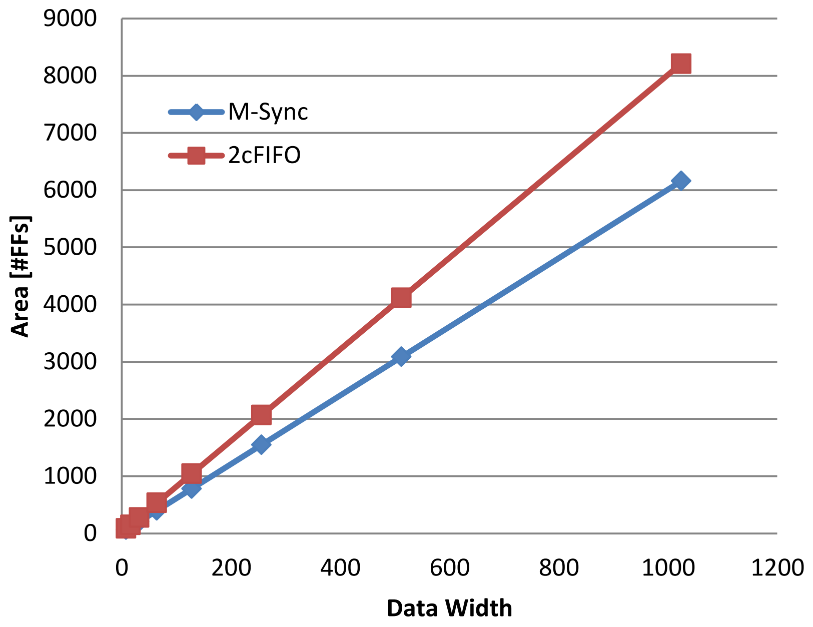


Figure 18: Area Comparison

5. CONCLUSIONS

In this paper data transfer between mesochronous clock domains was discussed. A new four-stage mesochronous synchronizer with back-pressure and burst support was presented. The paper shows that a minimal depth of four buffers is required to avoid timing violations, when a real reset circuitry is considered. This is in contrast with previous publications that employed three buffers. The four-stage mesochronous synchronizer provides low latency and full throughput, while requiring less area than a standard two-clock FIFO. The four-stage mesochronous synchronizer can be easily configured to support both long range source synchronous communication and long-wires between the transmitter and receiver sides of the synchronizer. In addition, suitable synchronizer configurations can also deal with clock phase drift and custom burst length. The four-stage mesochronous synchronizer relies only on synthesizable logic and therefore it is suitable for any SoC, FPGA and NoC applications.

REFERENCES

ALSHAIKH, M., KINNIMENT, D. and YAKOVLEV, A. 2010. Robust Synchronization using the Wagging Technique. Technical Report. TR NCL EECE-MSD-TR-2010-165.

CAPUTA, P. and SVENSSON, C. 2006. An On-Chip Delay- and Skew-Insensitive Multicycle Communication Scheme. In *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC 2006),* San Francisco, U.S, 1765-1774.

CUMMINGS, C.E., 2002. Simulation and Synthesis Techniques for Asynchronous FIFO Design. SNUG 2002 . User Papers.

DALLY, W.J. AND POULTON, J.W. 1998. *Digital Systems Engineering*. Cambridge UniversityPress.

DOBKIN, R. and GINOSAR, R. 2008. Fast Universal Synchronizer. In *Proceedings of the 18th International Workshop PATMOS,* Lisbon, Portugal, 199-208.

DOBKIN, R. and GINOSAR, R. 2009. Two Phase Synchronization with Sub-cycle Latency. *VLSI Integration Journal, 42(3) ,* 367-375.

FRANK, U., KAPSCHITZ, T. and GINOSAR, R. 2004. *Formal Methods in System Design (special issue on Formal Methods for Globally Asynchronous Locally Synchronous Design), 28(2),* 171-186.

FRIEDMAN, E.G. 2001. Clock Distribution Networks in Synchronous Digital Integrated Circuits. In *Proceedings of the IEEE 89(5),* 665-692.

GHONEIMA, M., ISMAIL, Y., KHELLAH, M. and DE, V. 2007. Variation-Tolerant and Low-Power Source-Synchronous Multi-Cycle On-Chip Interconnection Scheme. *IEEE Trans. on Circuits and Systems I, Regular Paper, vol. 2007,* 1-12.

GINOSAR, R. 2005. MTBF of a MultiSynchronizer System on Chip. In http://webee.technion.ac.il/~ran/papers/MTBFmultiSyncSoc.pdf.

JEX, J. and DIKE, C. 1995. A fast resolving BiNMOS synchronizer for parallel processor interconnect. *IEEE Journal of Solid-State Circuits, 30(2),* 133-139.

KOL, R. and GINOSAR, R., 1998. Adaptive Synchronization, In *Proc. IEEE International Conference on Computer Design* (ICCD 1998).

KOL, R. and GINOSAR, R. 2000. Adaptive Synchronization. In *Proceedings* of *Asynchronous Interfaces Workshop (AINT'2000)*, TU Delft, The Netherlands, 93-101.

LUDOVICI, D., STRANO, A. *et al.,* 2009. Comparing Tightly and Loosely Coupled Mesochronous Synchronizers in a NoC Switch Architecture. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOC2009),* Salo, Finland, 244-249.

MENG, T.H. 1990. *Synchronization Design of Digital Systems.* . Kluwer Academic Publishers.

MESGARZADEH, B. and SVENSSON, C. 2002. A New Mesochronous Clocking Scheme for Synchronization in SoC. In *Proceedings of 2004 IEEE International Symposium on Circuits and Systems (ISCAS'04),* Vancouver, Canada, 605-609.

MU, F. and SVENSSON, C. 2001. Self-Tested Self-Synchronization Circuit for Mesochronous Clocking. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, Vol.48, no.2,* 129-141.

SAPONARA, S., VITULLO, F. *et al.,* 2008. LIME: A Low-latency and Low-complexity On-chip Mesochronous Link with Integrated Flow Control. In *Proceedings of the 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD),* Parma, Italy, 32-35.

SEMIAT, Y. and GINOSAR, R. 2003. Timing Measurements of Synchronization Circuits. In *Proceedings of the 9th IEEE Internetional Symposium on Asynchronous Circuits and Systems (ASYNC'03),* Vancouver, Canada, 68-77.

SPARSO, J. and FURBER, S. 2001. *Principles of asynchronous circuit design - A systems perspective*. Kluwer Academic Publishers.

VANGAL, S.R., HOWARD, J. *et al.,* 2008. An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *IEEE Journal of Solid-State Circuits, 43(1),* 29-41.

VITULLO, F., L'INSALATA, N.E. *et al.*, 2008. Low-Complexity Link Microarchitecture for Mesochronous Communication in Networks-on-Chip. *IEEE Trans. on Computers, Vol.57,no.9,* 1196-1201.

WIKLUND, D. 2003. Mesochronous Clocking and Communication in On-Chip Networks. In *Proceedings of the Swedish System-on-Chip Conference (SSoCC'03).*