Fast Universal Synchronizers

Rostislav (Reuven) Dobkin[1] and Ran Ginosar[1]

[1] VLSI Systems Research Center, Electrical Engineering Department,
Technion – Israel Institute of Technology,
38200 Haifa, Israel
rostikd@tx.technion.ac.il, ran@ee.technion.ac.il

Abstract. Synchronization circuits are essential in multi-clock-domain systems-on-chip. The most well-known synchronizer consists of two sequentially connected flip-flops that should eliminate the propagation of metastability into the receiver clock domain. We first clarify how such a simple "two-flop" synchronizer can be used in the system, and analyze its performance, showing that the data cycle may be as long as 12 clock cycles. Novel faster synchronizers are described next and their use and improved performance are explained. The fast synchronizer enable shorter data cycles, measuring only 2 to 4 clock cycles. Synchronizer performance is also analyzed when the two communicating clock domains are separated by long interconnect, incurring additional latencies.

Keywords: Synchronization, MCD, SoC

1 Introduction

Systems on chip (SoC) typically integrate multiple modules that may operate at different clock frequencies, constituting multiple clock domain (MCD) devices. Multiple clock domains may be required either due to different external frequencies, or the integration of modules that were designed to operate on different frequencies, or to facilitate clock gating and partitioning of large and fast clock trees. In addition, frequency and voltage may also be changed dynamically in Dynamic Voltage and Frequency Scaling (DVFS) systems [1]-[3], mainly to reduce power consumption.

The mutual relationships of pairs of clock domains are classified in Table 1 according to the frequency and phase differences of the two domains. Mesochronous domains share the same frequency and have a constant phase difference between them, which can be compensated by relatively simple synchronizers [4][5], e.g. by a small FIFO. Adaptive phase compensation can be employed to connect multi-synchronous domains, in which the phase drifts slowly over time [6][7], as well as plesiochronous domains [8], where a very small frequency difference can be viewed as a phase drift. When two different-frequency clocks are used in the periodic case, a predictive synchronizer foresees and prevents contentions [9]. In the general asynchronous case, when the timing of input is unknown, the family of two flip-flop ("two-flop") synchronizers and two-clock FIFOs are employed. In addition, more

Table 1: Clock relationship classes

Class	$\Delta\phi$	Δf	Synchronization
Synchronous	0	0	None
Mesochronous	ϕ_C	0	Phase compensation
Multi-synchronous	drifts	0	Adaptive phase compensation
Plesiochronous	varies	$\Delta f<\varepsilon$	Adaptive phase compensation
Periodic		$\Delta f>\varepsilon$	Predictive
Asynchronous			Two-Flop, 2-clock FIFO

complex low-latency synchronizers that employ stoppable and locally-delayed clocks are also applicable for the asynchronous case [10]-[17]. They must take into account additional latency due to clock tree delays [17]-[19], may require non-standard gates, incur timing assumptions and may be restricted to a certain range of clock rates. Therefore, some applications must resort to the family of two-flop synchronizers, discussed in this paper. The synchronizers for the asynchronous class are universal because they also support all other classes. However, universal synchronizers do not take advantage of knowing the clock relationships and hence they are sub-optimal for other classes, incurring performance overhead.

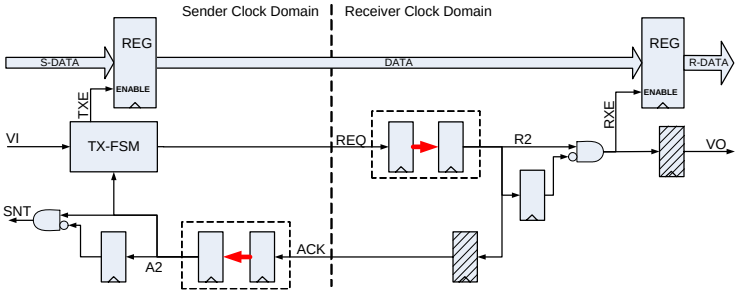

Fig. 1: Simple four-phase synchronizer

Synchronizers should be employed carefully, matching system requirements in terms of rate, latency and reliability, and avoiding common pitfalls [20]. A simple synchronizer is shown in Fig. 1. The flip-flops sampling the asynchronous signals REQ and ACK may become metastable. One clock cycle is preserved for the metastability resolution, and no logic is allowed on the bold (red) arrows of Fig. 1. The exact time required for single synchronizer metastability resolution is derived from system Mean Time Between Failures (MTBF) requirement [21]-[23]. The MTBF of single synchronizer is calculated according to Eq. (1), where S is the time preserved for metastability resolution, τ and W are technology dependent constants (about one and two FO4 inverter delays respectively) and F_C and F_D are clock and data frequencies respectively. When the time required for metastability resolution is longer than one clock cycle, additional flip-flops can be inserted before first flip-flop. Alternatively, when the requirement is shorter than one half clock cycle, a first falling edge flip-flop can be used. The receiver may also employ a READY signal, pausing the synchronizer when the receiver is not ready to receive (ACK signal is not returned until READY becomes high). In Fig. 1 we consider a simple version, which assumes

that the receiver is always ready. In this case the cross-lined flip-flops can be omitted, reducing the data cycle by two receiver clocks. Additionally, the data register at the RX side may also be omitted.

$$MTBF = \frac{e^{S/\tau}}{W \cdot F_C \cdot F_D} \tag{1}$$

The transmitter FSM and overall STG [24] are shown in Fig. 2. Note that '+' indicates a rising edge and '-' denotes a falling edge. The REQ is sent after input valid indication VI, provided that the synchronizer has finished its previous cycle (A2 is low). Output valid VO is pulsed for one RX cycle after a new data word has been received and synchronized, and sent indication SNT is pulsed for one TX cycle after A2 is received.

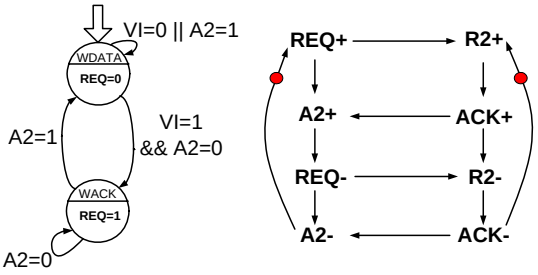

Fig. 2. Simple Synchronizer: Sender FSM and synchronizer STG

The simple synchronizer enables reliable communication between two clock domains. Unfortunately, that synchronizer is limited to low data rates. In typical cases of mutually-asynchronous clocks, 6 TX cycles and 6 RX cycles are required for a complete and acknowledged transfer of a single word.

Two-clock FIFOs can be employed for throughput enhancement enabling data transfer on each clock cycle. However, the FIFO is a more complex design that incurs higher data latency and does not support communications over long interconnects. In this paper we present novel faster synchronizers, especially in the presence of long wires between the transmitter and the receiver. We consider four- and two-phase protocols in Sect. 2 and 3 respectively, and compare their performance with that of the two-clock FIFO in Sect. 4. The synchronizer latency can be improved further by sampling multiple times and employing speculative or non-speculative voting [25][26].

2 Fast Four-Phase Synchronizer

The main goal of the synchronizer is to provide sufficient time for metastability resolution for the first sampling flip-flop. The resolving time of the synchronizer should meet MTBF requirements; in Fig. 3 we show a fast four-phase synchronizer, which provides one clock cycle for metastability resolution. Actually, the exact time reserved for metastability resolution in Fig. 3 is:

$$T_{RESOLUTION}^{M/S} = T - T_{SU}^{ENABLE} \qquad (2)$$

where T is the cycle time of the sampling clock. A similar caution should be applied to the AND gates in the figure.

When fast clocks are used either in the transmitter, or in the receiver, or both, a single cycle time may be insufficient for reliable operation. In this case the time for metastability resolution can be extended by inserting additional flip-flops as shown in Fig. 3. For finer latency optimization (e.g. when additional half cycle is required) one can employ flip-flops triggered by the falling edge of the clock. Alternatively, when a clock is slow, the time for metastability resolution can be reduced by clocking the ACK and REQ sample registers with the falling edge. As above, the output of the resolving flip-flop is marked in bold (red). These lines require special treatment to allow for sufficient resolution time. They should not be combined with other parts of the logic. While other logic may be synthesized normally, caution should be applied to avoid manipulation of the bold (red) lines by the logic synthesizer and physical design software. In particular, note that certain registers have two separate enable inputs: one normal and one marked in red and asterisk, which cannot be simply merged by logic.

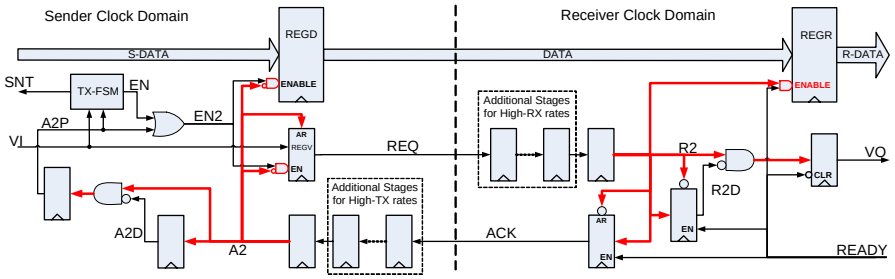

Fig. 3: A fast four-phase synchronizer

The operation of the synchronizer is explained by means of the STG in Fig. 4a and TX FSM in Fig. 4b. At the beginning, the synchronizer waits for data which is indicated by rising VI. The transmitter output registers (REGD and REGV) are enabled and will send out the new data word and REQ on the next rising edge of TX clock. At the receiver side, if the receiver is ready (READY is high), DATA is sampled by REGR and a VO pulse is generated after R2 rises. Timing of this event depends on metastability resolution and may be delayed by an extra clock cycle. Note that metastability of the first sampling flip-flop can only result in non-determinism in timing, and R2 is not expected to assume an illegal voltage level (except, maybe, once per MTBF…). The receiver then produces the rising ACK signal. Once sampled, A2 disables TX output registers (REGD and REGV) and asynchronously resets REQ. The output registers stay disabled until the four-phase REQ/ACK handshake is over. The falling edge of R2 triggers an asynchronous de-assertion of ACK. Following the synchronized falling edge of ACK, the transmitter enables the next data cycle once a new data word is available.

As shown in Fig. 5a, in mesochronous operation, the minimal data cycle time (REQ+ → REQ+) is six clock cycles in the worst case (the two clocks are in phase) but only four clock cycles when the two clocks are out of phase (Fig. 5b).

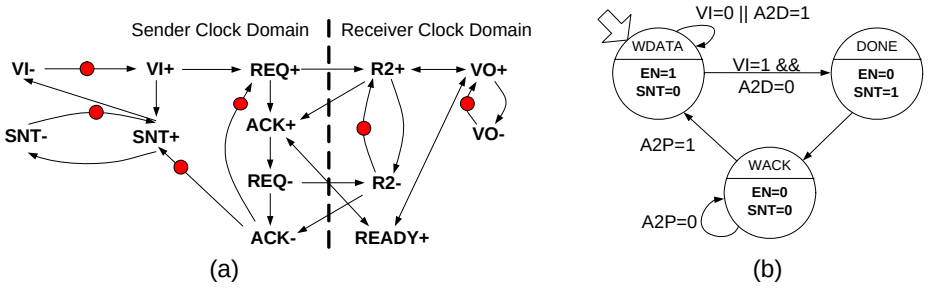

Fig. 4. Fast four-phase synchronizer STG

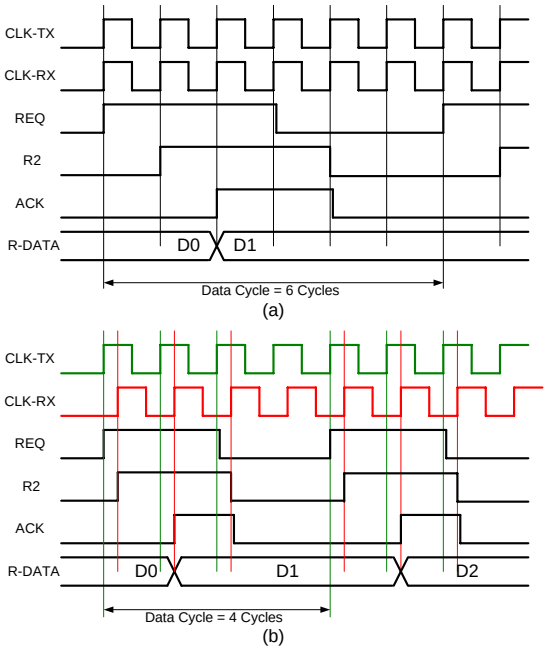

Fig. 5: Mesochronous operation of the fast four-phase synchronizer:
(a) in phase clocks (b) off-phase clocks.

The synchronizer of Fig. 3 supports any relation between the transmitter and receiver clocks. When the clocks are mutually asynchronous then the data cycle depends largely on the slower clock. If the ratio is larger than 2, then the data cycle is less than three clock cycles of the slower clock.

3 Fast Two-Phase Synchronizer

The synchronization data-rate can be significantly improved by employing a two-phase protocol over the channel. This is particularly important for long range commu-

nication where wires incur additional high latency.

When a two-phase protocol is employed, the synchronizer requires additional control logic. In the circuit shown in Fig. 6 ACK generation is symmetric for ACK+ and ACK- (no asynchronous resets).

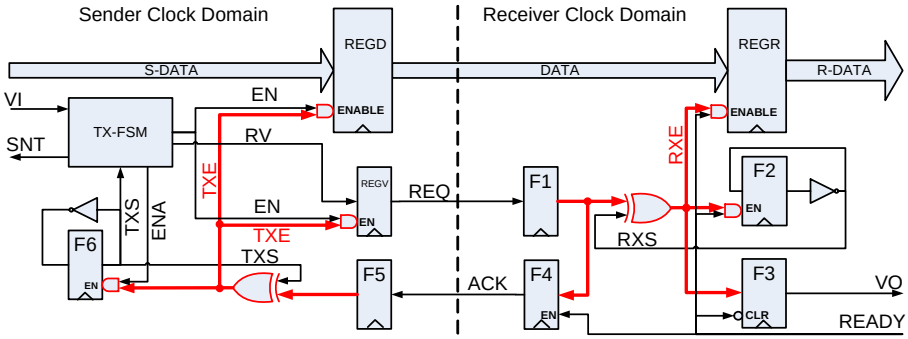

Fig. 6: Fast two-phase synchronizer

The synchronizer operation is explained by means of the STG in Fig. 7 and the TX FSM in Fig. 8. The time reserved for metastability resolution is shorter than in the four-phase synchronizer (Sect. 2) by the XOR gate delay. Note that TXS (the TX state) is produced by the (bold, red) synchronization circuit and hence, its toggle time depends on metastability resolution. The TX FSM accommodates this variability of toggling time. The output registers REGD and REGV are controlled by the FSM and by TXE (TX enable) the resolving signal from the sampling flip-flop marked in bold and red.

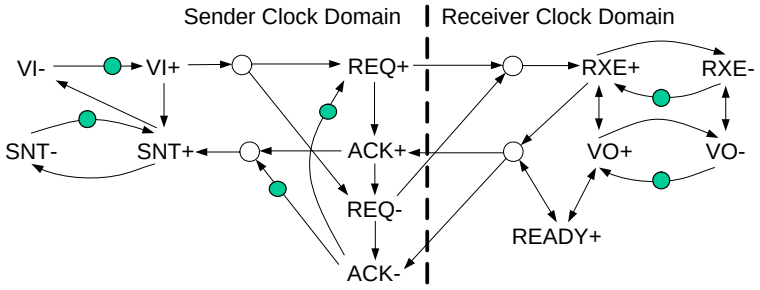

Fig. 7. STG of the fast two-phase synchronizer

In mesochronous operation, the minimal data cycle time (REQ+ → REQ+) is four clock cycles in the worst case (the two clocks are in phase), as shown in Fig. 9a. This data cycle is shorter (three clock cycles) when the clocks are out of phase (Fig. 9b). Note that the value of the non-zero phase difference in Fig. 9b has no impact on the data cycle.

The synchronizer of Fig. 6 supports any timing relationship between the transmitter and receiver clocks. When the two clocks are asynchronous then the data cycle depends largely on the slower clock. In particular, Fig. 10 shows a data cycle of mere two clock cycles when the frequency ratio is larger than two.

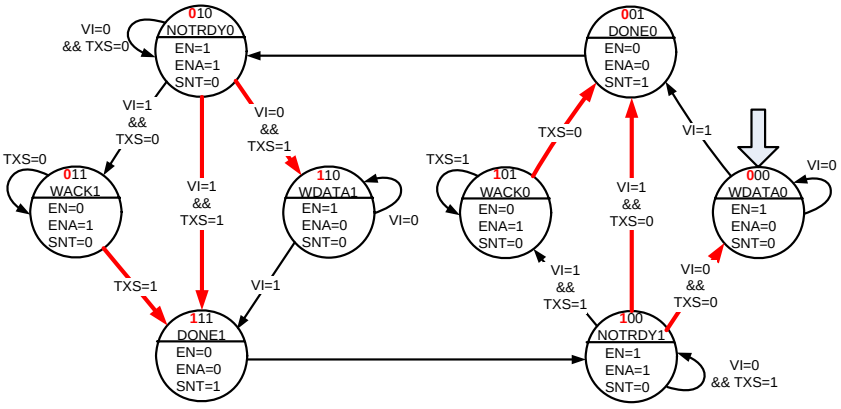

Fig. 8: TX FSM of the fast two-phase synchronizer

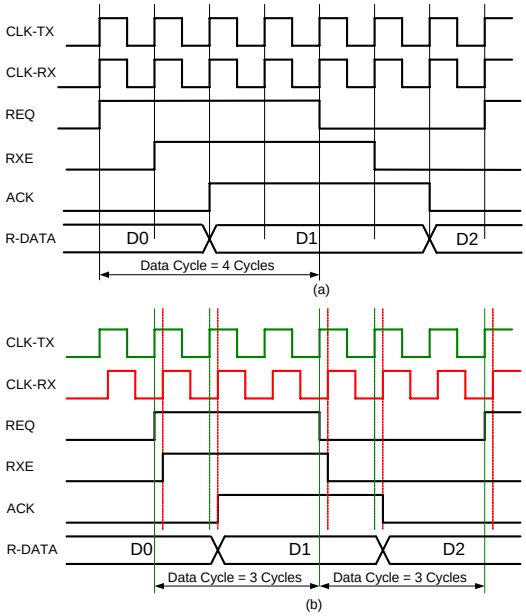

Fig. 9: Mesochronous operation of the fast two-phase synchronizer:
(a) clocks in phase (b) off-phase clocks

4 Performance Comparison

The goal of the synchronizers presented in this paper is to enhance throughput and latency; power and area of the synchronizers are immaterial, because only a tiny fraction of total power and area in typical SoCs are consumed by synchronizers. The

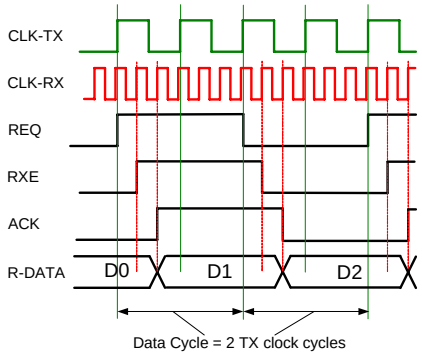

Fig. 10: Asynchronous clock domains. One clock is three times faster leading to data cycle of two clock cycles of the slower clock

lower bound of the data rate of any synchronizer that employs REQ/ACK handshake is two clock cycles, since at least one cycle is required for metastability resolution on either side. When a four-phase protocol is used, the lower bound is doubled up to four cycles.

The performance of the various synchronizers shown in this paper is summarized in Table 2. The simple synchronizer requires 12 cycles for each data transfer. When one of the clocks is faster, the data cycle will converge down to six cycles of the slower clock. The data cycles of the fast synchronizers are significantly improved down to four cycles in the case of fast two-phase synchronizer. The data cycle can be further reduced down to two clock cycles of the slower clock when the two clocks differ significantly in frequency.

The data rate and latency of the synchronizers over a range of clock ratios and two different interconnect delays between the transmitter and receiver are shown in Fig. 11 and 13. The results were obtained by simulations for worst case clock relations.

Forward latency refers to the average time from asserting REQ to asserting VO. Fig. 11 reflects back-to-back clock domains (no delay over the interconnect) and Fig. 12 shows an interconnect delay of one TX clock cycle. The two-clock FIFO is applicable only in the former case [27]. The fast two-phase synchronizer achieves only half the throughput of the FIFO, and the other two synchronizers provide even slower data rates. Forward latencies of the simple and fast synchronizers are shorter than the FIFO's. In Fig. 12, the throughput and latency depend linearly on the interconnect delay. Clearly, the fast two-phase synchronizer achieves that best throughput and latency.

Table 2: Four-phase and two-phase synchronizers latencies

	Simple Four-Phase		Fast Four-Phase		Fast Two-phase	
	Best (off-phase)	Worst (in-phase)	Best (off-phase)	Worst (in-phase)	Best (off-phase)	Worst (in-phase)
Mesochronous Clocks	10	12	4	6	3	4
Asynchronous Clocks	6·TX+6·RX		3·TX+3·RX		2·TX+2·RX	

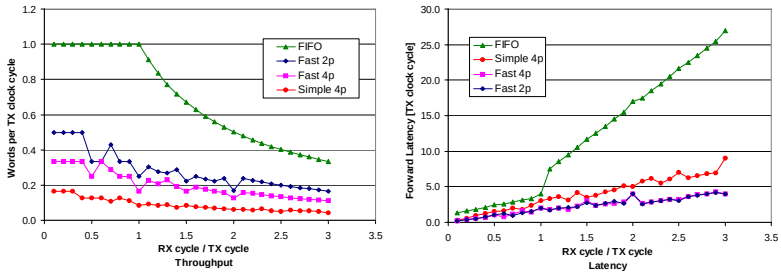

Fig. 11: Throughput and latency for no inter-modular delay

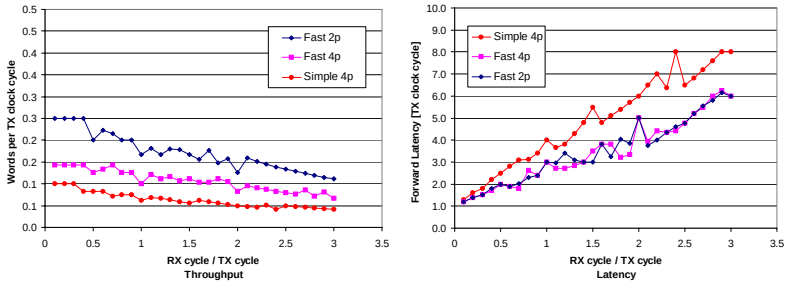

Fig. 12: Throughput and Latency for interconnect delay of 1.0 TX clock cycle

4 Conclusions

Synchronizers must be employed when transferring data across clock domain boundaries. Typical synchronizers may incur a heavy performance penalty. We have analyzed the structure and performance of simple two-flop synchronizer, and shown that its data cycle can take as long as 12 clock cycles. We have then presented novel faster designs that are based on either four-phase or two-phase protocols. The improved synchronizers can operate as fast as two clock cycles in certain cases. This improvement is accentuated when the communicating clock domains are far away from each other, and the delays on the interconnecting lines need to be taken into account; the paper presents a novel analysis of synchronizer behavior in the presence of long wire delays, and introduces two-phase synchronizers for minimizing the latency of such synchronizers.

References

[1] Semeraro G., Albonesi D.H., Dropsho S.G., Magklis G., Dwarkadas S., Scott M.L.: Dynamic Frequency and Voltage Control for a Multiple Clock Domain Microarchitecture. IEEE/ACM Int. Symp. on Microarchitecture, 356--367 (2002)

[2] Nielsen L. S., Niessen C., Sparsø J., van Berkel C. H.: Low-power Operation Using Self-timed and Adaptive Scaling of the Supply Voltage. IEEE Transactions on VLSI Systems, 2(4), 391--397 (1994)
[3] Daasch W.R., Lim C.H., Cai G.: Design of VLSI CMOS Circuits Under Thermal Constraint. IEEE Transactions on VLSI Systems, 49(8), 589--593 (2002)
[4] Dally W. J., Poulton J. W.: Digital System, Engineering (Eds.). Cambridge University Press (1998)
[5] Meng T. H.-Y.: Synchronization Design for Digital Systems(Eds.). Kluwer Academic Publishers (1991)
[6] Ginosar R., Kol R.: Adaptive Synchronization. ICCD, 188—189 (1998)
[7] Semiat Y., Ginosar R.: Timing Measurements of Synchronization Circuits. ASYNC, 68-77, 2003.
[8] Dennison L.R., Dally W.J., Xanthopoulos D.: Low-latency Plesiochronous Data Retiming. Advanced Research in VLSI, 304--315 (1995)
[9] Frank U., Kapschitz T., Ginosar R.: A Predictive Synchronizer for Periodic Clock Domains. J. Formal Methods in System Design, 28(2), 171--186 (2006)
[10] Kessels J., Peeters A., Wielage P., Kim S.J.: Clock Synchronization through Handshake Signaling. ASYNC, 59--68 (2002)
[11] Moore S., Taylor G., Mullins R., Robinson P.: Point to Point GALS Interconnect. ASYNC, 69--75 (2002)
[12] Oetiker S., Gürkaynak F.K., Villiger T., Kaeslin H., Felber N., Fichtner W.: Design Flow for a 3-Million Transistor GALS Test Chip. ACiD workshop (2003)
[13] Villiger T., Kaeslin H., Gürkaynak F.K., Oetiker S., Fichtner W.: Self-Timed Ring for Globally-Asynchronous Locally-Synchronous Systems. ASYNC, 141--150 (2003)
[14] Muttersbach J., Villiger T., Fichtner W.: Practical Design of Globally-Asynchronous Locally-Synchronous Systems. ASYNC, 52--61 (2000)
[15] Yun K. Y., Donohue R.P.: Pausible clocking: a first step toward heterogeneous systems. ICCD, 118--123 (1996)
[16] Yun K. Y., Donohue R.P.: Pausible clocking-based heterogeneous systems. TVLSI, 7(4), 482--488 (1999)
[17] Dobkin R., Ginosar R., Sotiriou C. P.: High Rate Data Synchronization in GALS SoCs. TVLSI, 14(10), 1063--1074 (2006)
[18] Sjogren A. E., Myers C. J.: Interfacing Synchronous and Asynchronous Modules within a High-Speed Pipeline. TVLSI, 8(5), 573--583 (2000)
[19] Mekie J., Chakraborty S., Sharma D.K.: Evaluation of Pausible Clocking for Interfacing High Speed IP Cores in GALS Framework. VLSI Design, 559--564 (2004)
[20] Ginosar R.: Fourteen Ways to Fool Your Synchronizer. ASYNC, 89--96 (2003)
[21] Dike C., Burton E.: Miller and Noise Effects in a Synchronizing Flip-flop. IEEE Journal of Solid-State Circuits, 34(6), 849--855 (1999)
[22] Ginosar R.: MTBF of Multi-synchronizer SoC. www.ee.technion.ac.il/~ran/papers/MTBFmultiSyncSoc.pdf.
[23] Kinniment D.J., Bystrov A., Yakovlev A.: Synchronization Circuit Performance. JSSC 37(2), 202--209 (2002)
[24] Chu T.A., Leung C.K.C., Wanuga T.S.: A Design Methodology for Concurrent VLSI Systems. Proc. of ICCD, 407--410 (1985)
[25] Kinniment D.J., Yakovlev A.: Low Latency Synchronization Through Speculation. PATMOS, 278--288 (2004)
[26] Kim S.J., Lee J.G., Kim K.: A Parallel Flop Synchronizer for Bridging Asynchronous Clock Domains. AP-ASIC, 184--187 (2004)
[27] Synopsys Design Ware FIFO, www.synopsys.com/products/designware/docs/doc/dwf/datasheets/dw_fifo_s2_sf.pdf.