High Rate Data Synchronization in GALS SoCs

Rostislav (Reuven) Dobkin, Ran Ginosar, and Christos P. Sotiriou

***Abstract*—Globally asynchronous, locally synchronous (GALS) systems-on-chip (SoCs) may be prone to synchronization failures if the delay of their locally-generated clock tree is not considered. This paper presents an in-depth analysis of the problem and proposes a novel solution. The problem is analyzed considering the magnitude of clock tree delays, the cycle times of the GALS module, and the complexity of the asynchronous interface controllers using a timed signal transition graph (STG) approach. In some cases, the problem can be solved by extracting all the delays and verifying whether the system is susceptible to metastability. In other cases, when high data bandwidth is not required, matched-delay asynchronous ports may be employed. A novel architecture for synchronizing inter-modular communications in GALS, based on locally delayed latching (LDL), is described. LDL synchronization does not require pausable clocking, is insensitive to clock tree delays, and supports high data rates. It replaces complex global timing constraints with simpler localized ones. Three different LDL ports are presented. The risk of metastability in the synchronizer is analyzed in a technology-independent manner.**

***Index Terms*—Asynchronous circuits, globally asynchronous, locally synchronous (GALS), synchronization, system-on-chip (SoC).**

I. Introduction

As SILICON technology continues to make rapid progress, systems-on-chip (SoCs) incorporate an increasing number of modules of growing sizes, operating at faster clock frequencies. These developments make it even more difficult to distribute a single synchronous clock to the entire chip [1]. As an alternative, different methods for providing each module with its own clock are being developed. Another motivation for independently clocking different modules, is to reduce power consumption by means of dynamic voltage and frequency scaling (DVFS) [2]–[4]. When the clock frequencies of the various modules are uncorrelated with each other, and when they can change over time independent of the clocks of other modules, the resulting SoC is termed a globally asynchronous, locally synchronous (GALS) system [5], [6]. Each GALS module (a "Locally Synchronous Island") can be enclosed in an asynchronous wrapper (Fig. 1), which facilitates inter-modular communications and generates the clock for the module [7]–[14].

Data synchronization and communication across clock domains in GALS architectures constitute a major challenge. The simple "two-flop" synchronizer typically incurs significant multicycle latency and limits the throughput. An alternative solution is provided by elastic first-in, first-out (FIFO) buffers. The most promising approach employs stoppable or stretchable clocks for the GALS modules: Port controllers (Fig. 1) can pause the local clock when sampling asynchronous input. By stopping the local GALS clock during data transfers across clock domains, the possibility of metastability is eliminated [7]–[14].

This paper offers two main contributions. First, we demonstrate that GALS systems employing pausable clocks are subject to failures, resulting from delays in their clock distribution networks. We propose modifications to the existing stoppable clock method, which mitigate these problems and yield robust GALS circuits. The modifications require post-layout verification of certain constraints on the clock network delays, to assure safe GALS clocking.

Second, we describe a novel synchronization technique for GALS SoC, locally delayed latching (LDL). It does not require pausable clocking, it is insensitive to clock tree delays, and it provides high data rate synchronization. We also propose a performance enhancement of LDL over its previous version [15]. Detailed reliability analysis for LDL is presented. In addition, we present a number of possible GALS wrappers based on LDL.

The paper begins with a survey of related research, in Section II. Standard GALS clocking and synchronization are reviewed and analyzed in Section III. Their potential for failure and exact failure conditions of the GALS approach are demonstrated in Section IV. In Section V, modifications to the conventional GALS approach are presented that mitigate failures. In Section VI, we present LDL synchronization and analyze its failure probability in a technology independent manner. LDL simulation results are discussed in Section VII.

II. Related Work

Two principal clocking and synchronization methods have been proposed for solution of the data synchronization problem. Clock synchronization employs handshake clocks that are stopped based on inputs from other domains [7]. Stoppable local clocks have been proposed in [8]–[13]. According to that methodology, a local ring-oscillator clock generator in each synchronous "island" incorporates a set of mutual exclusion elements (MUTEXes) [16] that stop the clock temporarily when new input data arrives, so as to avoid the risk of metastability. We analyze this methodology in detail in Section III.

Stoppable clocks have been introduced for GALS system in [12] and [13]. Asynchronous inter-modular communication is decoupled from the stoppable clock interface of the synchronous modules. A modified two-phase asynchronous interface wrapper for communication between two locally synchronous modules is presented in [8]. The authors also propose FIFO buffering for performance enhancement. A

Manuscript received June 11, 2005; revised February 17, 2006. This work was supported in part by ACiD-WG.

R. Dobkin and R. Ginosar are with the Electrical Engineering Department, Technion-Israel Institute of Technology, Haifa 32000, Israel (e-mail: rostikd@tx.technion.ac.il; ran@ee.technion.ac.il).

C. P. Sotiriou is with the Computer Architecture and VLSI Systems Division, ICS-FORTH, Vassilika Vouton, Heraklion, Crete, GR 711 10, Greece (e-mail: sotiriou@ics.forth.gr).

Digital Object Identifier 10.1109/TVLSI.2006.884148

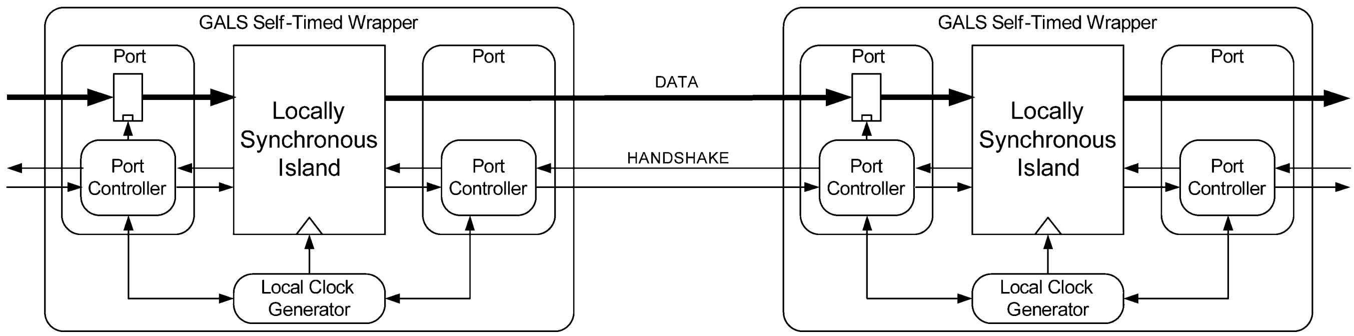


Fig. 1. GALS system [9].

four-phase version of the asynchronous GALS wrapper, which handles multiple ports and also facilitates testing, is presented in [9]. A number of GALS interconnect structures and modified wrappers are analyzed in [10], focusing on ring topology and packet based communications. An architecture for combining synchronous and asynchronous modules in a GALS system is presented in [11], employing handshake based on matched delays. Finite-state machine (FSM)-based demand and poll port controllers are also presented.

A stoppable clock technique for GALS pipelines [14], which does not employ MUTEX arbiters, accounts for clock tree delays by means of delay matching, and relies on accurate timing analysis of the clock tree. This solution is only suitable for linear pipelines and does not generalize to arbitrary GALS SoC communications.

In [17], a mixed-timing FIFO was proposed for communication between arbitrary combinations of synchronous and asynchronous domains. Mixed timing relay stations were also introduced for more efficient treatment of long interconnects. Source-synchronous communication, based on a self-timed single-stage FIFO with a single stage for mesochronous clock domains was presented in [18] and expanded to multisynchronous, plesiochronous, and asynchronous cases in [19]. The extensions are more complex relative to the mesochronous case, requiring additional special treatment at the transmitter and receiver sides.

In [20], abstract timing diagrams are used for analyzing synchronization interfaces, instead of the signal transition graph (STG) analysis [21] employed in [15]. In addition, [20] and [22] propose a new interfacing scheme, with a pausable clock generator at the transmitter side and free-running clock and partial handshake on the receiver side. Unfortunately, the authors report in [20] that without a biased MUTEX (a MUTEX that prioritizes one of the inputs, which is in practice physically unrealizable) the scheme may lead to "erroneous communication and loss of messages" when the communicating modules are not perfectly synchronized (in terms of data throughput). Reference [20] proposes a FIFO to partially solve that problem, whereby the clock is paused on the transmitter side when the FIFO is full. A detailed circuit at the transistor-level is presented in [22]. The circuit is reported to have an unknown nonzero failure probability.

III. Synchronization in Locally-Clocked GALS SoC

Clocking and input synchronization circuits for locally-clocked SoC proposed in the literature [8]–[13] are

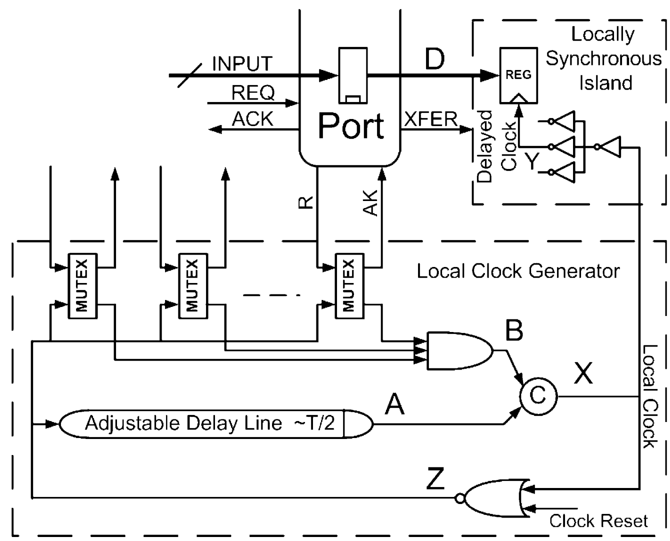


Fig. 2. Stoppable clock generation [8]

mostly variations of the circuit in Fig. 2. A locally generated pausable clock is employed in each Locally Synchronous Island. Input and output to other islands are controlled through asynchronous handshake on special ports. In this section, we analyze the operation of the circuit in Fig. 2.

The wrapper circuit operates as follows. Data arrival is indicated by input request REQ signal. In response, the port produces signal R, asking the local clock generator for a clock pause. Once the clock is paused, AK is asserted, enabling data latching by the port latch. After the data is latched, the port deasserts R, enabling the clock to the Locally Synchronous Island, in general, and to register REG in particular. Since the data (D) is stable by that time, it is assured that REG samples D correctly. At this point, input acknowledge ACK signal can be asserted, releasing the input handshake (REQ, ACK). Alternatively, ACK can be asserted right after data latching.

The clock generator [23] comprises a ring oscillator (consisting of the adjustable delay line, the NOR gate, and the C-element) and an arbitration circuit (Fig. 2). The arbitration circuit employs mutual exclusion elements (MUTEX, Fig. 3) [16]. The MUTEX grants G_i in response to request R_i, and guarantees that even if both requests arrive simultaneously, only one of them is granted. This is achieved by means of the metastability filter appended to the latch in the MUTEX. Five different signals related to the clock are shown in Fig. 2: A, B, X, Y, and Z. Each incoming request signal (REQ) is presented to the MUTEX (R),

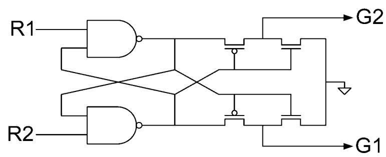


Fig. 3. MUTEX.

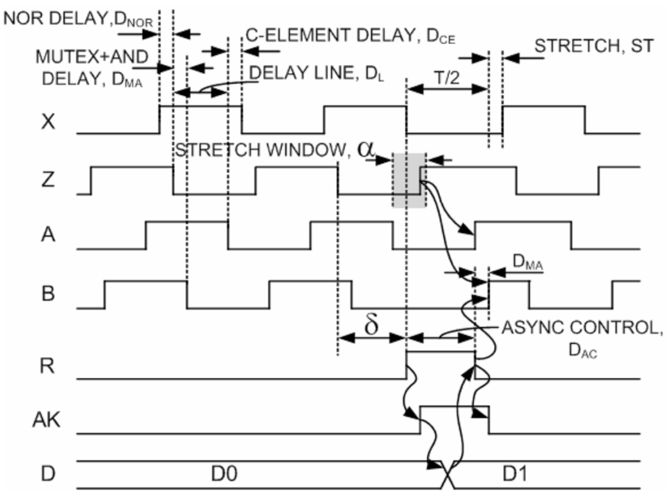


Fig. 4. Waveforms of stoppable clock generation.

asking for a clock pause. The MUTEX decides whether to grant the request (AK) or to allow the next clock pulse. The next clock pulse will take place only if all MUTEXes allow it, i.e., node B goes high. Let's denote the rising and falling transitions of a node n by $n+$ and $n-$, respectively. The C-element allows for the local clock signal X to be stretched ($X+$ transition is blocked) whenever any of the incoming R requests is granted (B is low) when A is rising. $X+$ will remain blocked until all granted R requests are released (and B eventually goes high).

The timing diagram in Fig. 4 illustrates the stoppable clock generation process. $R+$ is enabled in the MUTEX only when signal Z is low. The clock cycle is stretched when R arrives during a stretch window α, towards the end of the low phase of signal Z. If $R+$ arrives outside the stretch window, port handshake may complete on time ($B+$ precedes $A+$), causing no stretch.

The stretching process can also be described formally with a timed signal transition graph (STG) (Fig. 5), where the arcs are numbered for identification, and the labels on the arcs indicate symbolic transition delays. The STG is a special type of a Petri Net [21]. Tokens are marked by solid circles and their position (marking) determines the circuit state; the token marking in Fig. 5, denotes the initial state. Change of state is denoted by moving tokens along directed edges. A transition of node n is enabled when every incoming arc holds a token. When the transition takes place (node n "fires"), all incoming tokens are consumed and new tokens are produced on each outgoing arc. Places (marked by open circles) hold tokens in transit. It is assumed that every arc has a place for holding a passing token, but places are eliminated from the figure when there is no ambiguity. Place p1 is a choice place: The token can exit on either arc (13 or 14) but not on both, representing the free (random) choice made by the MUTEX in case of contention between $Z+$ and $R+$. Place p2 is a merge: It merges tokens arriving on either arc (19 or 20, depending on the previous choice) into arc (21).

The symbolic transition delays ($D_{\mathrm{MA}}, D_{\mathrm{CE}}, D_L, D_{\mathrm{NOR}}, D_{\mathrm{AC}}$) are defined in Fig. 4. The dashed arc labeled δ designates the delay from $Z-$ to $R+$. The timing of $R+$ may make the δ arc part of the critical path in the circuit, stretching the clock. Δ_{CLK} denotes clock tree delay from $X+$ to $Y+$, and ξ denotes the delay from incoming $R+$ until the data latching event D.

The stretch-length (ST in Fig. 4) can assume either a deterministic or nondeterministic value. ST is deterministic when there is no contention between R and Z signals at any MUTEX input. In case of contention, the MUTEX incurs an additional nondeterministic delay $T_{m/s}^{\mathrm{MUTEX}}$, causing the stretch-length ST to become nondeterministic too. The contention happens when R and Z both rise within a danger window W, typically three to four gate-delays long.

Let $\delta' \in [0,T)$ be the time from $Z+$ to $R+$. Since $R+$ is ignored when $Z=1$, we define δ as the effective time from $Z-$ to a port request, as follows:

$$\delta = \begin{cases} \delta' + \frac{T}{2}, & 0 \le \delta' \le W \\ 0, & W < \delta' \le \frac{T}{2} \\ \delta' - (\frac{T}{2}), & \frac{T}{2} < \delta' < T. \end{cases} \tag{1}$$

Note that $0 \le \delta < T/2 + W$. From Fig. 5 it can be observed that a stretch occurs if the path $6 \rightarrow 23 \rightarrow 15 \rightarrow 16 \rightarrow 17 \rightarrow 19 \rightarrow 21 \rightarrow 13 \rightarrow 4$ takes longer than a clock cycle

$$\delta + D_{\mathrm{AC}} + D_{\mathrm{MA}} + D_{\mathrm{CE}} + D_{\mathrm{NOR}} > T. \tag{2}$$

Note that in case of contention at a MUTEX input, D_{AC} takes $T_{m/s}^{\mathrm{MUTEX}}$ longer than in a noncontending case (arc 15 timing is extended by an additional $T_{m/s}^{\mathrm{MUTEX}}$ delay, see Fig. 5).

Stated otherwise, we have two sets of stretch conditions

$$T - (D_{\mathrm{AC}} + D_{\mathrm{MA}} + D_{\mathrm{CE}} + D_{\mathrm{NOR}}) < \delta < \frac{T}{2} - W \tag{3}$$

$$\frac{T}{2} - W \le \delta < \frac{T}{2} + W \tag{4}$$

When inequality (3) holds, the stretch is deterministic. In the other case [inequality (4)], it is nondeterministic. Subtracting the lower bound of (3) from the upper bound of (4), we obtain α, the size of the stretch window

$$\alpha = (D_{\mathrm{AC}} + D_{\mathrm{MA}} + D_{\mathrm{CE}} + D_{\mathrm{NOR}}) + W - \frac{T}{2}. \tag{5}$$

The stretch length ST, is the difference between the two sides of inequality (2)

$$\mathrm{ST} = (\delta + D_{\mathrm{AC}} + D_{\mathrm{MA}} + D_{\mathrm{CE}} + D_{\mathrm{NOR}}) - T. \tag{6}$$

Note that in the case of contention at a MUTEX input the stretch is extended by the delay of metastability resolution $T_{m/s}^{\mathrm{MUTEX}}$ (and the delay D_{AC} becomes nondeterministic). Combining (5) and (6) and separating the contention case from

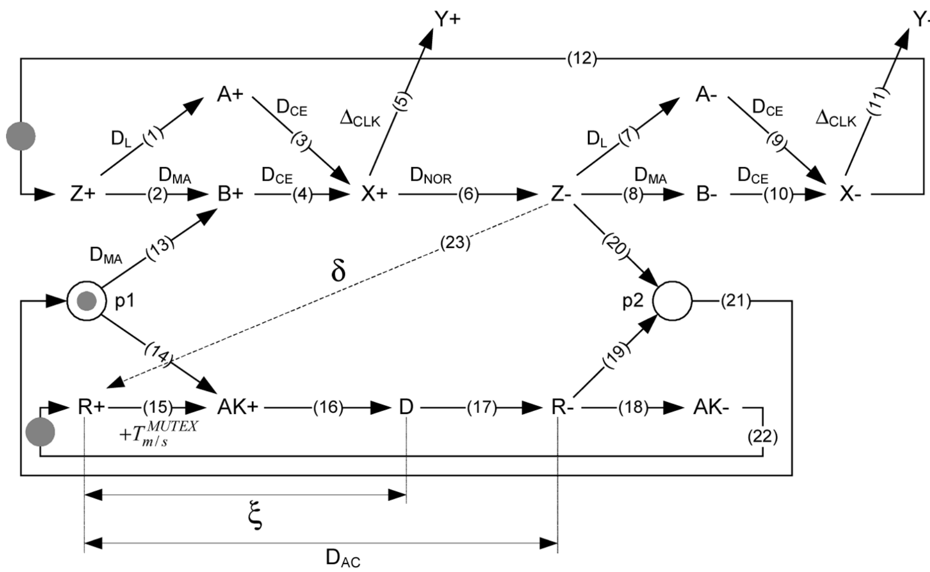


Fig. 5. Timed STG of the local stoppable clock of Fig. 2.

the noncontending case, we get the following expression for the stretch:

$$\mathrm{ST} = \begin{cases} \delta + \alpha - W - \frac{T}{2}, \\ T - (D_{\mathrm{AC}} + D_{\mathrm{MA}} + D_{\mathrm{CE}} + D_{\mathrm{NOR}}) \leq \delta < \frac{T}{2} - W \\ \delta + \alpha - W - \frac{T}{2} + T_{m/s}^{\mathrm{MUTEX}}, \frac{T}{2} - W \leq \delta \leq \frac{T}{2} + W. \end{cases} \tag{7}$$

If the clock cycle is relatively long, inequality (3) becomes infeasible. However, contention is still possible if $R+$ happens within $\pm W$ of $Z+$.

Fortunately, $T_{m/s}^{\mathrm{MUTEX}}$ can be bounded for any practical application as explained in Section VI-B. Therefore, when a relatively long clock cycle is employed, the stretched probability becomes insignificant.

Arc 23 represents only the unknown delay δ between the clock and the asynchronous request. It does not represent any causal relationship between them, and it is not an essential arc in the STG. Indeed, an infinite amount of tokens may accumulate on arc 23, as a result of a free running clock when no REQ arrives. All other arcs can accommodate, at most, one token at a time.

IV. Synchronization Failures in a Locally-Clocked GALS SoC

The approach described in Section III disregards the delay Δ_{CLK} along the clock tree (from node X to Y), thus, potentially causing metastability events in the sampling register REG, of the Locally Synchronous Island (Fig. 2). A failure scenario is detailed in Section IV-A and analyzed in Section IV-B.

A. Clock Delay Failure

A failure caused by the clock tree delay is depicted in Fig. 6. Let's assume that a request comes with delay δ after $Z-$ and

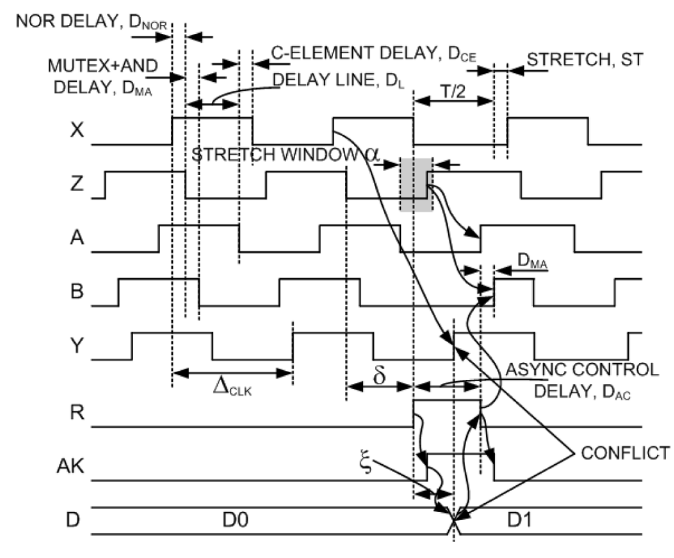


Fig. 6. Conflict example.

is granted by the MUTEX. Being uncorrelated with the input handshake, the delayed Clock Y may rise simultaneously with the asynchronous data latching in the Port. This conflict can cause metastability in the input REG of the Synchronous Island. ξ in Fig. 6 denotes the circuit delay from $R+$ to latching data in the port (shown also in STG of Fig. 5). In addition to the internal asynchronous port delay, ξ comprises the MUTEX delay, which may become larger in case of concurrent $R+$ and $Z+$ as explained in the previous section. Note that, even though Fig. 6 presents the conflict during a stretched cycle, the conflict may happen also when no stretch of the clock occurs, since the events $\mathrm{AK}+$ and $Y+$ are uncorrelated.

In addition to the metastability problem, this approach suffers from two other drawbacks, i.e., pausing the local clock slows down the entire Synchronous Island, and the slowdown may be

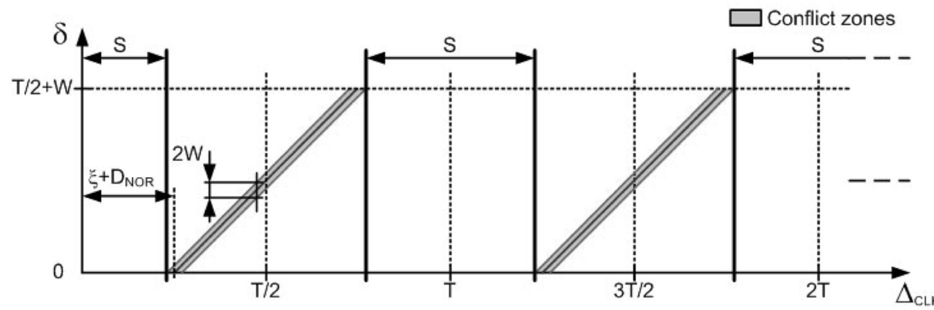


Fig. 7. Conflict zones.

TABLE I
CLOCK TREE DELAYS-IMPLEMENTATION EXAMPLES, 0.18-μM TECHNOLOGY

Design	Clock Frequency	Clock Cycle, T	Clock Skew	Clock Tree Delay
DLX-SYNC (FF)	278 MHz	3.60 ns	60 ps	0.530 ns (15% of T)
DES (FF)	540 MHz	1.85 ns	180 ps	1.168 ns (64% of T)
AES (OpenCores)	350 MHz	2.85 ns	165 ps	1.111 ns (39% of T)
MEM Control (OpenCores)	200 MHz	5.00 ns	126 ps	1.014 ns (20% of T)
Dual Clock MEM Control (OpenCores)	200/100 MHz	5.00 ns	137 ps	1.016 ns (20% of T)

exacerbated with multiport GALS modules, where the probability of pausing the clock is higher. Slowing down those synchronous islands which are critical to system performance may slow down the entire system.

B. Conflict Analysis

In this section, we analyze the conditions required for a conflict event. Starting from $X+$, the conflict occurs when

$$\Delta_{\mathrm{CLK}} \approx \delta + \xi + D_{\mathrm{NOR}}. \tag{8}$$

Namely, when the delay along arcs $6 \rightarrow 23 \rightarrow 15 \rightarrow 16$ on the STG matches the delay along arc 5 in Fig. 5. More precisely, the conflict occurs when $Y+$ happens inside a "danger window" $2W$ (setup + hold time) around $\delta+\xi+D_{\mathrm{NOR}}+\mathrm{k}\cdot T$, where k is an integer ($k > 0$ accounts for clock delays longer than T). The δ value is unknown, but the probability of conflict grows with the number of GALS module ports. In addition, the nondeterministic delay ξ can be bounded for any practical implementation as explained in Section VI-B. Fig. 7 emphasizes graphically the combinations of Δ_{CLK} and δ that lead to conflicts. The graph represents the timing relationships of (8). Note that for some values of Δ_{CLK}, independent of δ, the probability of conflict is negligible ("safe" regions S in Fig. 7). Alternative solutions that avoid such conflicts are described in Section V.

Clock tree delay is a function of the clock tree depth and depends on both technology and architecture. In traditional synchronous circuit design the delay of the clock tree is immaterial, as the clock is constantly running, and only the skew is important. Clock tree delay is proportional to the number of sequential elements driven by the clock and to the tolerable skew. Clock skew balancing becomes increasingly difficult for high-performance large SoC designs, incurring higher clock tree delays. For instance, in a 0.18-μm technology, a typical clock frequency achievable with standard electronic design automation (EDA) tools and standard libraries is 100–500 MHz ($T = 2$–10 ns), while typical clock delays are 1–2 ns, depending on module size (some examples are presented in Table I). Large SoCs, with tens of modules, may require much longer clock delays, approaching T. With faster technologies and larger chips, $\Delta_{\mathrm{CLK}} > T$ may become common if a single global synchronous clock is attempted for the entire SoC. Thus, while $\delta \in [0, T/2 + W)$, the range of the clock tree delay is not limited by T.

V. METASTABILITY-FREE GALS CLOCKING

In this section, we present metastability-free circuits for robust data synchronization of GALS modules with stoppable clocking. In Section V-A, we propose an approach for verifying the correctness of the original circuit of Fig. 2. The approach is based on delays extracted from the layout of the original circuit. Section V-B proposes a modification of Fig. 2 that avoids metastability at the expense of performance. In Section VI, we introduce novel LDL approach for data synchronization.

A. Timed Clock Trees

In this section, we show how to verify whether the circuit of Fig. 2 performs correctly in a given chip. As we show in Fig. 7, negligible failure probability is expected for the values of Δ_{CLK} that fall inside safe regions S. The size and position of the safe regions depend on a variety of parameters, such as the clock cycle length, design library cell delays (e.g., NOR-gate delay) and asynchronous port delay. It may be possible to verify that a conflict probability is negligible by performing timing analysis of the physical design and verifying that Δ_{CLK} falls only inside the safe regions.

For example, let's consider the case of a limited delay clock tree, when $\Delta_{\mathrm{CLK}} \leq T$. We verify that either

$$\Delta_{\mathrm{CLK}} < D_{\mathrm{NOR}} + \xi - T_H \tag{9}$$

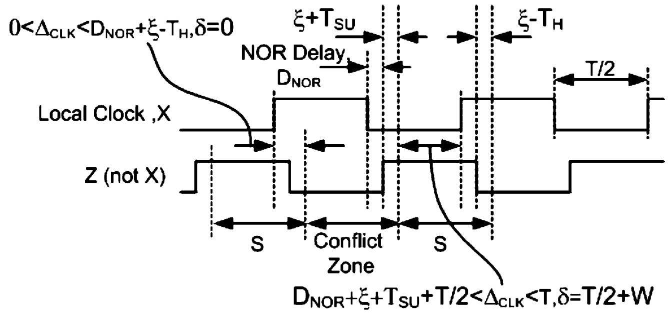


Fig. 8. Conflict and safe zones (example).

or

$$\Delta_{\text{CLK}} > D_{\text{NOR}} + \frac{T}{2} + \xi + T_{\text{SU}}. \tag{10}$$

T_{SU} and T_{H} are the setup and hold times of the DFF, respectively. When either rule holds, $Y+$ will occur only inside safe region S (Figs. 7 and 8). The first bound results from the $\delta = 0$ case and the second bound from the $\delta = T/2 + W$ case. Both cases relate to the $6 \rightarrow 23 \rightarrow 15 \rightarrow 16$ path in Fig. 5.

Similarly, when $\Delta_{\text{CLK}} > T$, the port access is allowed only during the S intervals (Figs. 7 and 8). In this case, either

$$\Delta_{\text{CLK}} < D_{\text{NOR}} + \xi - T_H. \tag{11}$$

Or the following two equalities must hold:

$$\begin{aligned} D_{\text{NOR}} + \frac{T}{2} + \xi + T_{\text{SU}} + k \cdot T &< \Delta_{\text{CLK}} \\ D_{\text{NOR}} + \xi - T_H + (k+1) \cdot T &> \Delta_{\text{CLK}} \end{aligned} \tag{12}$$

where $k = 0, 1, 2, \ldots$.

This solution has several drawbacks. The correctness constraints must be verified after each layout iteration of the circuit. The solution is not scalable and it may be sensitive to thermal and power supply voltage variations (different changes in ξ, T_{SU}, T_H, and D_{NOR}).

B. Matched Delay Port Control

An alternative solution to fulfilling the clock tree depth constraints presented above is to insert a delay line into the circuit of Fig. 2, thus, matching the clock-tree delay Δ_{CLK}, as shown in Fig. 9. By delaying the handshake, it can be guaranteed that it will always happen after the clock has been stopped.

However, the use of this matched delay may cause longer clock stretching, as demonstrated in Fig. 10, where in the worst case the stretch is additionally expanded by Δ_{CLK}. Note that the stretch window α is also expanded to α' (up to $T/2 + W$)

$$\alpha' = \begin{cases} \alpha + \Delta_{\text{CLK}}, & \Delta_{\text{CLK}} \leq \frac{T}{2} + W - \alpha \\ \frac{T}{2} + W, & \Delta_{\text{CLK}} > \frac{T}{2} + W - \alpha. \end{cases} \tag{13}$$

During the design process, it must be verified that the matched delay always exceeds Δ_{CLK}, over all possible process, voltage, and temperature (PVT) variations (namely, all corners and all in-die process variations). In addition, this type of solution is not

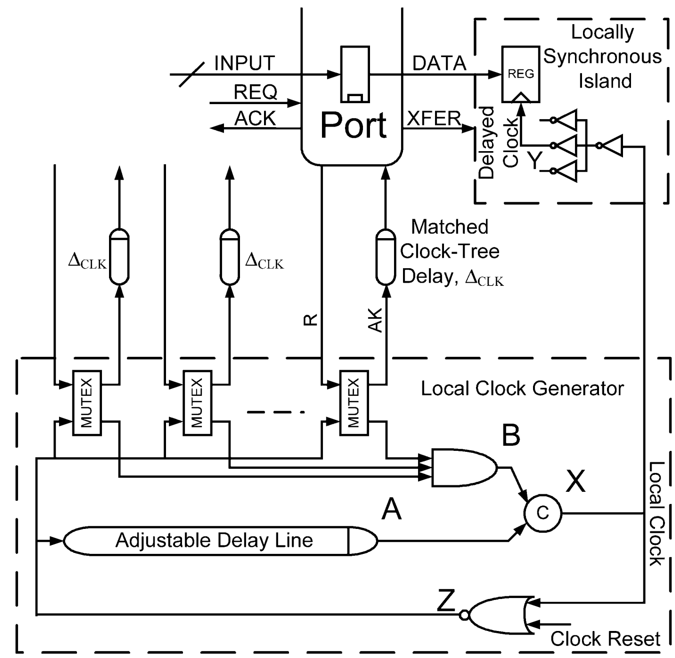


Fig. 9. Stoppable clock generation with matched clock-tree delays.

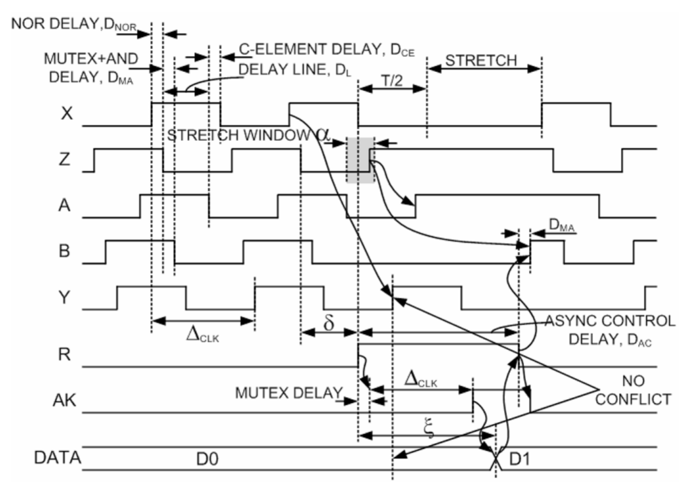


Fig. 10. Matched delay port control-wave diagram.

viable for designs with high clock rates, which often imply long clock tree delays. In such designs, clock stretch may happen on each handshake, since in this case it is very likely that $\alpha \rightarrow T/2$. In addition, this approach presents similar drawbacks as the "constrained delay clock tree."

VI. LDL Synchronization

In this section we introduce the LDL approach, which allows for GALS inter-modular communication and synchronization without the need for an arbitrated clock. It is shown that the LDL approach replaces the constraints on the clock delay (which were discussed in Section V) by simpler and more localized timing constraints, which are easier to achieve and verify. The LDL concept is described in Section VI-A, and the LDL tradeoff between reliability and data rate is analyzed in Section VI-B. LDL constraints are detailed in Section VI-C. A

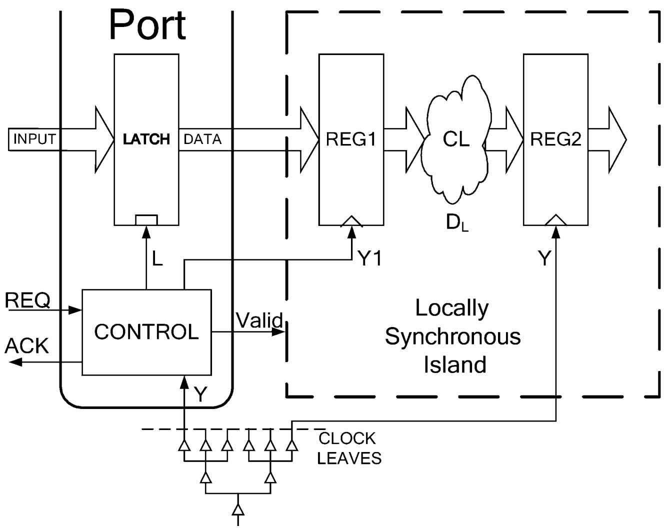


Fig. 11. LDL circuit.

further enhancement of LDL is described in Section VI-D, and expected performance is discussed in Section VI-E. Finally, in Section VI-F–VI-H, we provide three implementation examples of input and output ports. Section VII presents simulations of these implementations.

A. LDL Principles

In an LDL input port synchronizer, the asynchronous controller (Fig. 11) controls both the input latch and Y1, the clock input to the first sampling register. Signal Y, the local clock of the module, is uninterrupted. In addition, the port issues a valid indication for each newly received data word and prevents WRITE after READ (WAR) hazards. Various modes of the LDL operation are demonstrated in Fig. 12.

In LDL, the clock of the Locally Synchronous Island is never stopped. The only measure available is to delay $Y1+$ when a conflict is imminent. $Y1-$ is unaffected, and only the high-phase is shortened. A port request is accepted only during the low-phase of Y, latching the incoming data $(L+)$ and delaying $Y1+$ when needed. The conflicts between $Y+$ and $\mathrm{REQ}+$ are resolved by a MUTEX inside the control. A number of such asynchronous controllers for generating $(L+)$ and $Y1+$ are presented in Section VI-F–VI-H.

LDL is unaffected by clock cycle changes that can be caused for instance due to dynamic frequency or voltage scaling [2]–[4]. There is also no restriction on stopping the clock during periods of inactivity.

B. LDL Synchronization Reliability and Performance

The worst case operation occurs when at the conflict between $\mathrm{REQ}+$ and $Y+$, $\mathrm{REQ}+$ wins. In this case the high-phase of $Y1+$ is maximally shortened, shown in Fig. 12, in the "Port Wins" case. The shorter cycle leaves less time for computing in the combinational logic immediately following the first register. The implementation must assure that the remaining high phase is long enough, according to the restrictions on the minimal high phase width for FFs or registers of the target library. The high phase of the clock is shortened by an amount equal to the latency of the asynchronous control (D_{CTRL}) and the MUTEX resolution latency. To analyze this further, we define the following.

1) Resolution: A metastable MUTEX resolves when the value stored in its internal latch is set nondeterministically to either 1 or 0, and all combinational functions of that value (MUTEX outputs) have been evaluated. The resolution latency is indeterminate and unbounded.

2) Failure: A circuit is said to fail if a combinational function of the output of a metastable MUTEX of that circuit does not resolve within a predefined maximal time $T_{m/s}^{\mathrm{MUTEX}}$.

3) Safety: A circuit is M-safe if the expected time between two successive failures exceeds M [M is also known as mean time between failures (MTBF)] [24].

4) Min High Clock Phase: $T_{\mathrm{HP}}^{\mathrm{Min}}$ is a minimally allowed clock high-phase time for a FF (typically about three FO4 inverter gate delays).

We require that the SoC be at least M-safe, where a selected value for M could be 100 years (other values may also be used). To achieve that, the safety of each synchronizer in a SoC with about $K = 100$ synchronizers must be at least K times larger, namely $M = 10\,000$ years [25]. We note that, in a standard SoC (a digital IC based on standard cells and designed using standard EDA tools) the shortest clock cycle is typically about 100–160 FO4 inverter delays [26]. The nominal FO4 inverter delay depends mostly on the process technology (as detailed in Section VI-E). Thus, the fastest high phase (50% of the clock cycle) is about 50 inverter delays long. In order to assess the worst-case MTBF in the following equation, we assume that τ and W are one and two FO4 inverter delays, respectively, [24], $F_D = \mathrm{F_C}$ (worst-case analysis), the clock cycle $T = 100\,\tau$ and, thus, $F_C = F_D = 1/100\,\tau$. We can determine the required metastability resolution time $T_{m/s}^{\mathrm{MUTEX}}$ in terms of a number of gate delays N, by solving

$$\mathrm{MTBF} = \frac{e^{T_{m/s}^{\mathrm{MUTEX}}/\tau}}{W \cdot F_C \cdot F_D} = \frac{e^N}{2 \times \frac{1}{100} \times \frac{1}{100}} \times \tau \geqslant 10^4 \quad \text{year}. \tag{14}$$

For $10^{-11} < \tau < 10^{-10}$ s (the range of FO4 gate delays in present and foreseeable technologies, cf. Section VI-E)

$$41 < N \approx \ln\left(\frac{10^7}{\tau}\right) < 43. \tag{15}$$

For $T = 100\,\tau$, this implies that at least one half of a symmetric clock cycle should be allowed for resolution. For slower SoCs, e.g., where the fastest clock cycle is $160\,\tau$ [26], a quarter clock cycle suffices to achieve this MTBF. For most aggressive designs [such as high-speed processors or high-speed application-specified integrated circuit (ASIC) modules], where $10\tau < T < 50\,\tau$, a different approach based on multicycle resolution time or on multisynchronous clocking [27] is required.

C. LDL Constraints

As explained in Section VI-B, to guarantee minimal high clock phase $T_{\mathrm{HP}}^{\mathrm{Min}}$, we require that

$$\frac{T}{2} - D_{\mathrm{CTRL}} - T_{m/s}^{\mathrm{MUTEX}} > T_{\mathrm{HP}}^{\mathrm{Min}}. \tag{16}$$

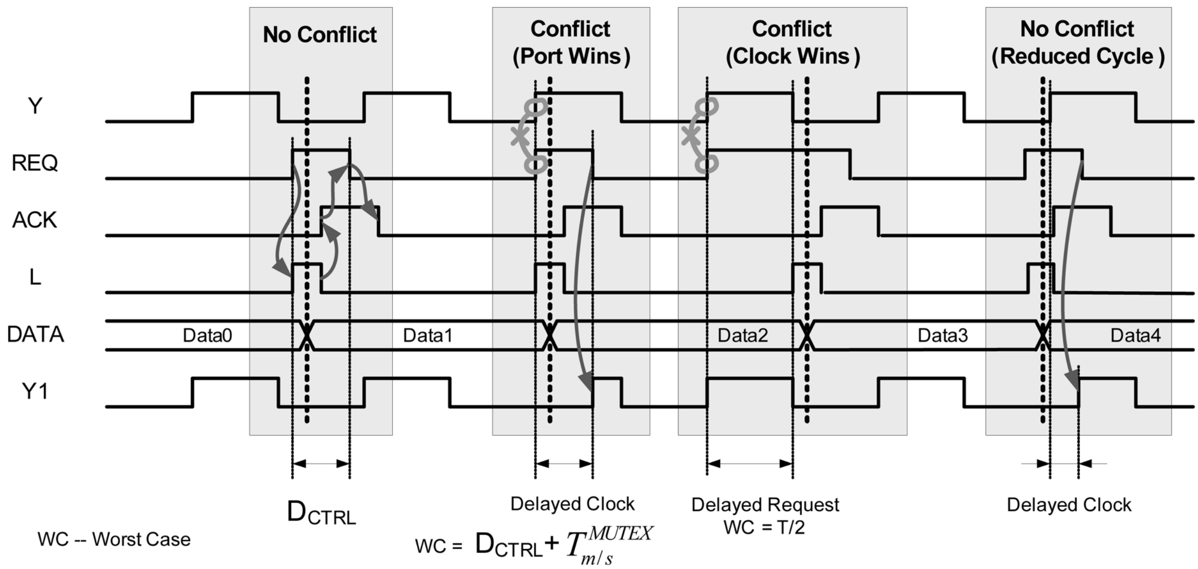


Fig. 12. LDL operating modes.

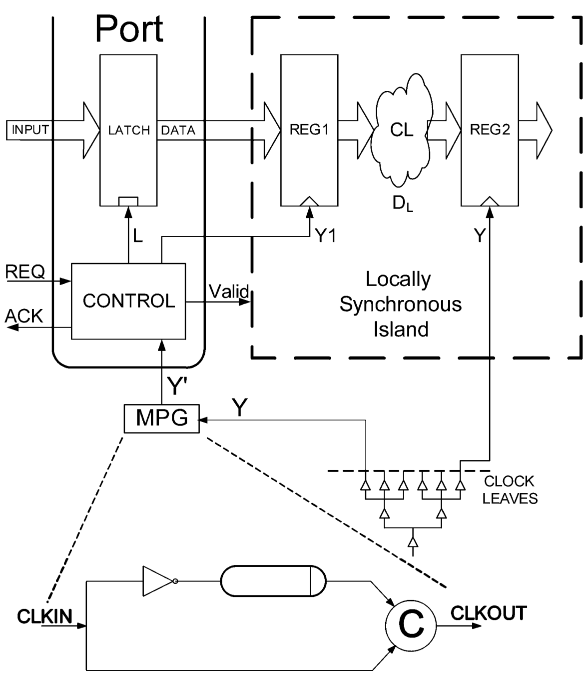


Fig. 13. Locally delayed latching circuit with MPG for an extended high phase.

Leading to a constraint on the asynchronous control delay

$$D_{\text{CTRL}} < \frac{T}{2} - T_{\text{HP}}^{\text{Min}} - T_{m/s}^{\text{MUTEX}}. \tag{17}$$

Another constraint applies to D_L, the delay of the combinational logic that follows REG1. When the rising edge of Y1 is delayed (by up to $D_{\text{CTRL}} + T_{m/s}^{\text{MUTEX}}$), the effective computation time in that logic stage becomes shorter than the clock cycle. Therefore, the following should be satisfied:

$$D_L < T - D_{\text{CTRL}} - T_{m/s}^{\text{MUTEX}}. \tag{18}$$

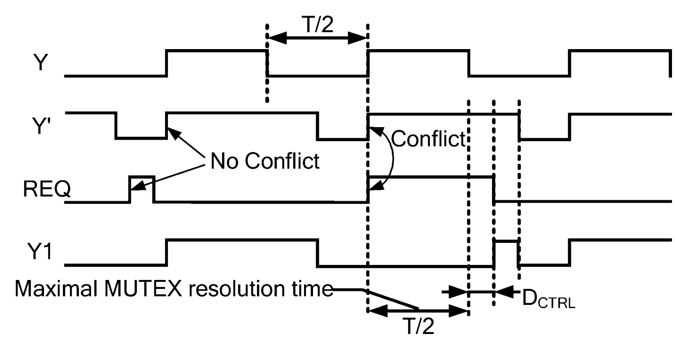


Fig. 14. Extended high phase for high frequency operation. After $\text{REQ} - Y'$conflict, REQ wins.

D_{CTRL} contains additional buffering delays when a wide data path is required. These constraints are verified in Section VII for the implementations in Sections VI-F–VI-H.

D. LDL Performance Enhancement

When a clock faster than $T = 160\tau$ is employed, MTBF requirements prohibit shortening the high phase of Y1. To circumvent that obstacle, a minimum phase generator (MPG) is employed as in Fig. 13 to guarantee that the high Y' clock phase is no shorter than $T_{m/s}^{\text{MUTEX}} + T_{\text{HP}}^{\text{Min}} + D_{\text{CTRL}}$ [see (16)]. For fast clocks, this minimum is longer than half a clock cycle. An example of a 75% duty cycle Y' clock is shown in Fig. 14. In this case, $Y1$ is also 75% duty cycle when there is no conflict. In time of conflict, there is sufficient margin in the high phase of Y' [at least 43 FO4 gate delays as in (15)], and $Y1$ is guaranteed to be no shorter than $T_{\text{HP}}^{\text{Min}}$. The request is treated during the low-phase period of Y'.

E. LDL Performance Versus Technology

Considering minimal low-phase and high-phase width of about three gate-delays each, asynchronous controller latency of about 20 gate-delays, and preserving 43 gate delays for metastability resolution [(15)], the minimal clock cycle is 69

TABLE II
MAXIMAL CLOCK FREQUENCIES ACHIEVABLE WITH LOCALLY DELAYED LATCHING

Technology	FO4 Gate Delay*	Maximal Rate (No MPG, Fig. 11)	Maximal Rate (With MPG, Fig. 13)
350 nm	95 ps	79 MHz	153 MHz
250 nm	67 ps	112 MHz	216 MHz
180 nm	47 ps	160 MHz	308 MHz
130 nm	33 ps	228 MHz	439 MHz
90 nm	22 ps	342 MHz	658 MHz
65 nm	15 ps	505 MHz	966 MHz

*Delays are based on High Performance processes [26]

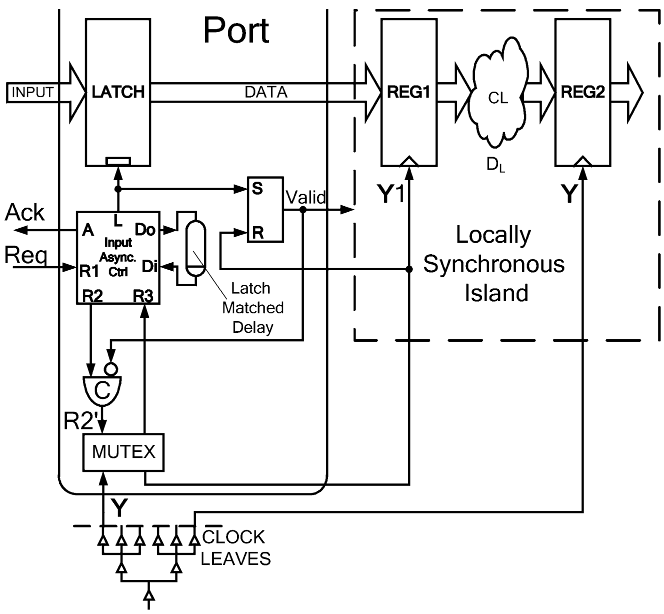


Fig. 15. GALS module decoupled input port.

gate-delays. Without the MPG enhancement of Section VI-D, when a 50% duty cycle clock is employed, the minimal clock half phase requires $3+20+43=66$ FO4 gate delays, namely the minimal clock cycle is 132 gate delays. The two alternative circuits enable clock frequencies up to the bounds shown in Table II. The table is based on scaled FO4 inverter delays, derived from the FO3 NAND delay model of the ITRS [26]. Similar results were reported also in [28] and [29]. This table applies to ASIC, and is inapplicable to high-speed microprocessors, where the cycle time is typically around 20 gate delays.

F. Decoupled Input Port

Fig. 15 shows a possible implementation of Fig. 11. Without a conflict, $Y1+$ is either not delayed or delayed by less than $D_{\rm CTRL}$. $R2'+$ is granted only during the low-phase of Y. The MUTEX arbitrates any conflict between $R2'+$ and $Y+$. When $R2'+$ wins over $Y+$, the asynchronous controller is granted ($R3+$). The controller employs an asymmetric matched delay Do $\rightarrow$ Di to open the latch and then close it again ($L+ \rightarrow L-$). The asymmetric delay applies a longer delay to one of the two edges. In our case, the delay is minimal for the falling edge and is longer for the rising one. The long delay matches the latch propagation delay for all corners and delay variations. After $R2'-, Y1+$ triggers REG1, leading to a shortened cycle in the combinational logic following REG1 (the cycle is shortened by

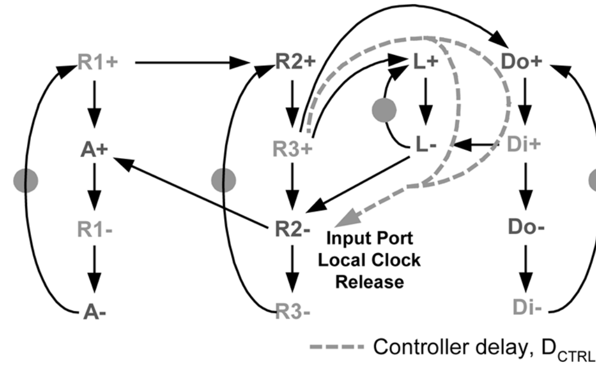


Fig. 16. GALS decoupled input port asynchronous control STG.

$D_{\rm CTRL}$). If the clock wins over $R2'+, R3+$ happens only half a cycle later, after $Y-$.

The STG of the decoupled input port control is shown in Fig. 16. The controller delay is measured along the dashed path. The path is contained entirely inside the input port, ensuring that any reduction of the clock cycle depends solely on the input port control logic (and not on the logic and clock of the transmitter module). The MUTEX output $Y1$ should be buffered with a low-skew net when wide data path is required. In this case, the additional latency must be taken into account. The latency of the controller is verified in Section VII.

Generation of the valid signal is performed as follows. For each granted data transfer the control issues signal L that latches the data and concurrently sets valid to logic high, indicating a new ready data word inside the data latch. Once latching is accomplished, the controller deasserts $R2$ ($R2-$) and along with valid + releases the MUTEX. The next incoming data transfer request ($R2+$) is blocked (by means of the c-element) until the data is sampled by REG1. Valid is released with the next rising edge of $Y1$, also enabling arbitration of the next incoming data transfer request. The SR latch of the valid signal is free of conflicts: L and $Y1$ are guaranteed to be mutually exclusive thanks to the MUTEX.

G. Decoupled Output Port

Output ports of GALS modules are subject to the same issues as the input ports which were discussed in the previous sections. The difference is that with an output port it is the incoming ACK signal which must be synchronized. An output port circuit is shown in Fig. 17 and the STG of its control is shown in Fig. 18. The internal acknowledge ($A1$) is decoupled from the external asynchronous handshake.

The controller latency of the decoupled output port control, $D_{\rm CTRL}^{\rm OUT}$, should be verified according to (17) and (18). In this case

$$D_{\rm CTRL}^{\rm OUT} = D[A4+ \rightarrow A1+ \rightarrow A3-]. \tag{19}$$

The latency of the controller is verified in Section VII.

H. Simpler Input Port

The architecture in Fig. 19 simplifies the input port of Section VI-F, eliminating the asynchronous controller from the input port.

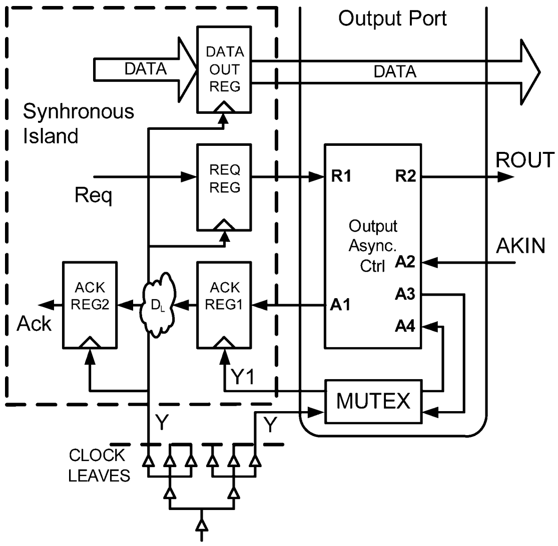


Fig. 17. GALS module decoupled output port.

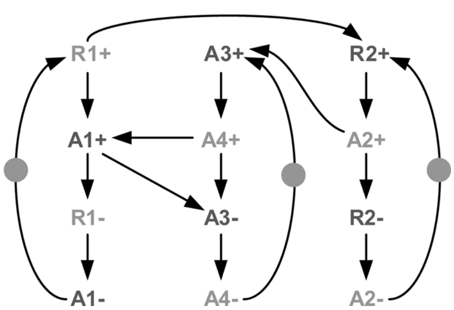


Fig. 18. GALS module decoupled output port asynchronous control STG.

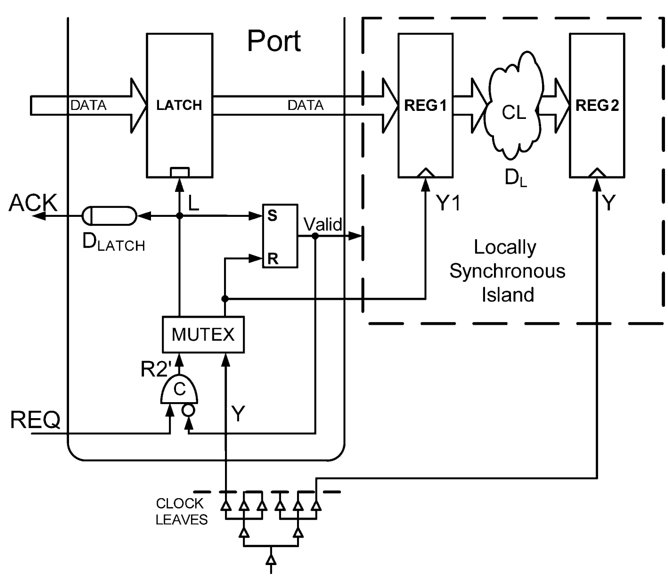


Fig. 19. Simple input port.

The input port delay D_{CTRL} now depends on the external delays of the output port

$$D_{\text{CTRL}} = D_{\text{LATCH}} + D_{\text{OutputPort}}(\text{ACK}+ \rightarrow \text{REQ}-). \tag{20}$$

The matched delay in Fig. 19 could have been reduced from D_{LATCH} to $D_{\text{LATCH}} - D_{\text{OutputPort}}$, but when $D_{\text{OutputPort}}$ is unknown *a priori*, it is better to leave the matched delay at D_{LATCH}. This simple input port is compatible with the output port of Section IV-G. The latency of this constellation (simple input port with the decoupled output port) is also verified in Section VII.

VII. Simulation

The circuits of Sections VI-F–VI-H were synthesized using Petrify [30], converted to VHDL, synthesized by the synopsys design compiler using 0.35- and 0.25-μm CMOS libraries [31], [32], and verified by gate level simulations with wire-load model delays (SDF). Table III lists the results for the three controllers. We employed 16-bit wide data buses and clock cycles of 160 FO4 gate delays in a standard ASIC [26].

According to (17), while one-quarter cycle is preserved for metastability resolution, we are left with another quarter cycle (40 inverter delays) for the delay of the asynchronous controller and for the clock high-phase. Our 0.35-μm library specifies $T_{\text{HP}}^{\text{Min}} = 0.361$ ns, namely about 3 inverter delays, leaving 37 gate delays for the asynchronous controller delay D_{CTRL}. Thus, we should verify that the delay of the circuits described in Sections VI-F–VI-H is less than 37 gate delays. The circuit delays are listed in Table III. All of them fulfill the delay requirement.

According to Table III, the delays of all three asynchronous controllers are lower than the bound of 37 FO4 gate delays, requiring roughly 10% 160 FO4 gate delays clock cycle. This margin allows operating at a slightly higher frequency, if needed. Evidently, this approach is limited by the time we reserve for the MUTEX to resolve. However, it provides a useful operational frequency range for most ASICs.

To preserve timing correctness, careful layout should be performed. The sampling latch, the first register REG1, the asynchronous control, and the MUTEX must be placed closely together in order to avoid the impact of wire propagation delay on the critical path. These requirements are expected to be met easily, since the wrapper contains only a single port and is not connected to any other parts of the module.

The overhead of the LDL controller is expected to be less than 100 gates (including the MPG). For example, the decoupled input port controller logic complexity is equivalent to 36 2-input NAND gates, and the MPG requires about 25 gates. For a typical SoC module of 100 K gates, the LDL controller overhead is only 0.1%. Another 0.1% overhead may be incurred by the latches of the input port.

VIII. Conclusion

We have addressed the problem of synchronization failures due to clock delays in locally generated, arbitrated clocks of GALS SoCs. The problem has been analyzed based on clock delays, cycle time, and complexity of the asynchronous port controllers. The analysis employs a timed STG approach in order to identify potential conflicts spanning asynchronous and synchronous circuits.

Several solutions have been discussed. First, we have shown that timing analysis can be used to verify known solutions (using

TABLE III
CONTROLLER DELAYS

Circuit	Critical Path	Latency (0.35µm)	Latency (0.25µm)	Num. of FO4 inverter delays	Estimated portion of 160τ clock cycle
Decoupled Input Port	R3+→Do+→Di+→L–→R2–	2.63 ns	2.10 ns	21	13 %
Decoupled Output Port	A4+→A1+→A3–	1.81 ns	1.22 ns	14	9 %
Simple Input Port with Decoupled Output Port	Latch Delay→A2+→R2–	2.14 ns	1.53 ns	16	10 %

(Note: Unlike Table II, our 0.35µm and 0.25µm library cells exhibit 130ps and 95ps FO4 gate delays, respectively)

arbitrated clocks) in the presence of clock delays. Second, a solution employing matched delays is described, where a control signal is delayed so as to match the clock delay and avoid synchronization failures.

A novel architecture for synchronizing inter-modular communications in GALS, based on LDL, has been presented. LDL synchronization does not require pausable clocking, is insensitive to clock tree delays, and supports high data rates. It replaces the complex global timing constraints on clock delays by simpler, more localized ones. Three different LDL ports have been described, two for input and one for output. Their operation has been demonstrated and analyzed by simulations. We also present a technology-independent analysis of the metastability risk in the synchronizer, and its effect on the synchronizer architecture.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for their many helpful and constructive comments.

REFERENCES

[1] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proc. IEEE*, vol. 89, no. 5, pp. 665–692, May 2001.

[2] G. Semeraro, D. H. Albonesi, S. G. Dropsho, G. Magklis, S. Dwarkadas, and M. L. Scott, "Dynamic frequency and voltage control for a multiple clock domain microarchitecture," in *Proc. IEEE/ACM Int. Symp. Microarch.*, 2002, pp. 356–367.

[3] L. S. Nielsen, C. Niessen, J. Sparsø, and C. H. van Berkel, "Low-power operation using self-timed and adaptive scaling of the supply voltage," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 391–397, Dec. 1994.

[4] W. R. Daasch, C. H. Lim, and G. Cai, "Design of VLSI CMOS circuits under thermal constraint," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 49, no. 8, pp. 589–593, Aug. 2002.

[5] D. M. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Comput. Sci. Dept., Stanford Univ., Palo Alto, CA, 1984.

[6] D. Bormann and P. Cheung, "Asynchronous wrapper for heterogeneous systems," in *Proc. ICCD*, 1997, pp. 307–314.

[7] J. Kessels, A. Peeters, P. Wielage, and S. J. Kim, "Clock synchronization through handshake signaling," in *Proc. ASYNC*, 2002, pp. 59–68.

[8] S. Moore, G. Taylor, R. Mullins, and P. Robinson, "Point to point GALS interconnect," in *Proc. ASYNC*, 2002, pp. 69–75.

[9] S. Oetiker, F. K. Gürkaynak, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, "Design flow for a 3-million transistor GALS test chip," in *Proc. ACiD Workshop*, 2003, ACiD workshop slides.

[10] T. Villiger, H. Kaeslin, F. K. Gürkaynak, S. Oetiker, and W. Fichtner, "Self-timed ring for globally-asynchronous locally-synchronous systems," in *Proc. ASYNC*, 2003, pp. 141–150.

[11] J. Muttersbach, T. Villiger, and W. Fichtner, "Practical design of globally-asynchronous locally-synchronous systems," in *Proc. ASYNC*, 2000, pp. 52–61.

[12] K. Y. Yun and R. P. Donohue, "Pausible clocking: A first step toward heterogeneous systems," in *Proc. Comput. Des.: VLSI Comput. Process., ICCD*, 1996, pp. 118–123.

[13] K. Y. Yun and R. P. Donohue, "Pausible clocking-based heterogeneous systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 4, pp. 482–488, Dec. 1999.

[14] A. E. Sjogren and C. J. Myers, "Interfacing synchronous and asynchronous modules within a high-speed pipeline," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 5, pp. 573–583, Oct. 2000.

[15] R. Dobkin, R. Ginosar, and C. Sotiriou, "Data synchronization issues in GALS SoCs," in *Proc. ASYNC*, 2004, pp. 170–179.

[16] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980, ch. 7.

[17] T. Chelcea and S. M. Nowick, "Robust interfaces for mixed-timing systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 8, pp. 857–873, Aug. 2004.

[18] A. Chakraborty and M. R. Greenstreet, "Efficient self-timed interfaces for crossing clock domains," in *Proc. ASYNC*, 2003, pp. 78–88.

[19] A. Chakraborty and M. R. Greenstreet, "A minimal source-synchronous interface," in *Proc. ASIC/SOC Conf.*, Sep. 2002, pp. 443–447.

[20] S. Chakraborty, J. Mekie, and D. K. Sharma, "Reasoning about synchronization techniques in GALS systems: A unified approach," in *Proc. Workshop Formal Methods GALS Arch. (FMGALS)*, 2003, invited paper.

[21] T. A. Chu, C. K. C. Leung, and T. S. Wanuga, "A design methodology for concurrent VLSI systems," in *Proc. ICCD*, 1985, pp. 407–410.

[22] J. Mekie, S. Chakraborty, and D. K. Sharma, "Evaluation of pausible clocking for interfacing high speed IP cores in GALS framework," in *Proc. VLSI Des.*, 2004, pp. 559–564.

[23] S. W. Moore, G. S. Taylor, P. A. Cunningham, R. D. Mullins, and P. Robinson, "Self-calibrating clocks for globally asynchronous locally synchronous systems," in *Proc. Int. Conf. Comput. Des. (ICCD)*, 2000, pp. 73–78.

[24] C. Dike and E. Burton, "Miller and noise effects in a synchronizing flip-flop," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 849–855, Jun. 1999.

[25] R. Ginosar, MTBF of multi-synchronizer SoC [Online]. Available: http://www.ee.technion.ac.il/~ran/papers/MTBFmultiSyncSoc.pdf

[26] *"International Technology Roadmap for Semiconductors (ITRS),"* (2003) [Online]. Available: www.itrs.net

[27] Y. Semiat and R. Ginosar, "iming measurements of synchronization circuits," in *Proc. ASYNC*, 2003, pp. 68–77.

[28] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[29] N. Weste and D. Harris*, CMOS VLSI Design—A Circuits and Systems Perspective*. Reading, MA: Addison-Wesley.

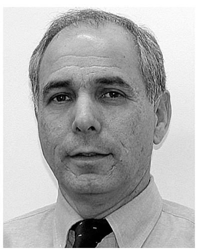
[30] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Trans. Inf. Syst.*, vol. E80-D, no. 3, pp. 315–325, 1997.

[31] J. B. Sulistyo, J. Perry, and D. S. Ha, "Developing standard cells for TSMC 0.25 μm technology under MOSIS DEEP rules," Dept. Elect. Comput. Eng., Virginia Tech., Blacksburg, Tech. Rep. VISC-2003-01, Nov. 2003.

[32] J. B. Sulistyo and D. S. Ha, "A new characterization method for delay and power dissipation of standard library cells," *VLSI Des.*, vol. 15, no. 3, pp. 667–678, 2002.

Rostislav (Reuven) Dobkin received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 1999 and 2003, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at the same institute.

From 1997 to 2000, he worked within RAFAEL ASIC Experts design group, Haifa, Israel, and between 2001–2002 he led a VLSI design group in IC4IC Ltd., Yokneam, Israel, developing family of chips for communications. In parallel, he served as a Teaching Assistant at Technion Electrical Engineering Department since 1999. His research interests include VLSI architectures, parallel architectures, asynchronous logic, high-speed interconnect, synchronization, globally-asynchronous, locally synchronous (GALS) systems, system-on-chip (SoC), and network-on-chip (NoC).



Ran Ginosar received the B.Sc. degree in electrical engineering and computer engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 1978, and the Ph.D. degree in electrical engineering and computer science from Princeton University, Princeton, NJ, in 1982.

He worked at AT&T Bell Laboratories, Murray Hill, NJ, from 1982 to 1983, and joined the Technion faculty in 1983. He was a visiting Associate Professor with the University of Utah, Salt Lake City, from 1989 to 1990, and a Visiting Faculty with Intel Research Labs, Hillsboro, OR, from 1997 to 1999. He co-founded four companies in the areas of electronic imaging, medical devices, and wireless communications. He serves as the head of the VLSI Systems Research Center at the Technion. His research interests include VLSI architecture, asynchronous logic, electronic imaging, and neuro-processors.

Christos P. Sotiriou received the Ph.D. degree in computer science from the University of Edinburgh, Edinburgh, Scotland, in 2001.

He is currently a Research Associate with the Institute of Computer Science (ICS) at the Foundation for Research and Technology-Hellas (FORTH), Crete, Greece, and a Visiting Assistant Professor at the University of Crete, Crete, Greece. His research interests include VLSI design, computer-aided design (CAD) algorithms and tools, and design methods and approaches for asynchronous timing. He served as General Chair of the International Symposium on Advanced Research in Asynchronous Circuits and Systems in 2004.

Dr. Sotiriou was a recipient of the Best Paper Award from the International Symposium on Advanced Research in Asynchronous Circuits and Systems.