Data Synchronization Issues in GALS SoCs

Rostislav (Reuven) Dobkin[1], Ran Ginosar[1] and Christos P. Sotiriou[2]
[1]VLSI Systems Research Center, Technion—Israel Institute of Technology, Haifa 32000, Israel
[2]ICS-FORTH, Crete, Greece

Abstract: Locally generated, arbitrated clocks for GALS SoCs [1] face the risk of synchronization failures if clock delays are not accounted for. The problem is analyzed based on clock delays, cycle times, and complexity of the asynchronous port controllers. A number of methods are presented. In some cases, it is sufficient to extract all the delays and verify whether the system is susceptible to metastability. In other cases, when high data bandwidth is not required, asynchronous synchronizers or matched-delay asynchronous ports may be employed. Arbitrated clocks may be traded off for locally delayed input and output ports, facilitating high data rates. The latter circuits have been simulated, to verify their performance.

Index Terms: Globally Asynchronous Locally Synchronous, clocks, synchronization failures.

1. Introduction

As systems on chip (SoC) become larger and faster, it is becoming increasingly difficult to distribute a single synchronous clock to the entire chip [2]. To overcome this problem, a Globally Asynchronous, Locally Synchronous (GALS) architecture has been proposed [3]. The ability to run at different frequencies (and different supply voltages) also contributes to power savings. Other reasons to adopt GALS methodology include the need to interface with multiple external clock domains. The principal challenge of GALS architectures is the need to synchronize data as it crosses different clock domains inside the SoC. Two principal clocking and synchronization methods have been proposed to address this issue. Clock synchronization employs handshake clocks that are stopped based on inputs from other domains [4]. Arbitrated locally generated clocks have been proposed in [1][5][6][7]. According to this methodology, a local ring-oscillator based clock generator in each synchronous "island" incorporates a set of MUTEX arbiters that stop the clock temporarily when new input data arrive. In this paper, we analyze the effect that the delays through the clock distribution networks might have on synchronization. We show that such delays, which were unaccounted for in previous publications, may lead to failures. We also present a variety of solutions to this problem.

Standard GALS clocking and synchronization is reviewed in Section 2, and their potential for failure is analyzed in Section 3. Proposed solutions are presented in Section 4 and the simulations of some of them are discussed in Section 5.

2. Synchronization in Locally-Clocked GALS SoC

Clocking and input synchronization circuits for locally-clocked SoC proposed in the literature [1][5][6] [7] are mostly variations of the circuit in Figure 1. Note the five different components of the clock, namely A, B, X, Y and Z.

A locally generated stoppable clock is employed in each Locally Synchronous Island. Input and output to other islands are controlled by asynchronous handshake via special ports. The clock generator comprises a ring oscillator with an adjustable delay line [8] and an arbitration circuit. Each incoming request for a clock pause (R) is connected to a MUTEX that decides whether to grant the request (AK) or to permit the next clock pulse. The next clock pulse will take place only if all MUTEXes allow it

---

For ease of reading the paper, it has been formatted in a single column and with large graphics, and hence it is longer than the upper bound of 10 pages. However, once formatted for the conference and once the figures are reduced, the paper will be made shorter than 10 pages, as required.

(node B high). Thanks to the C-element, local clock X may be stretched (X+ is blocked) whenever at least one of the incoming R requests is granted (B is low) while A is rising, and stretching will last until all granted R requests are released (and B goes high). The process is demonstrated in Figure 2.

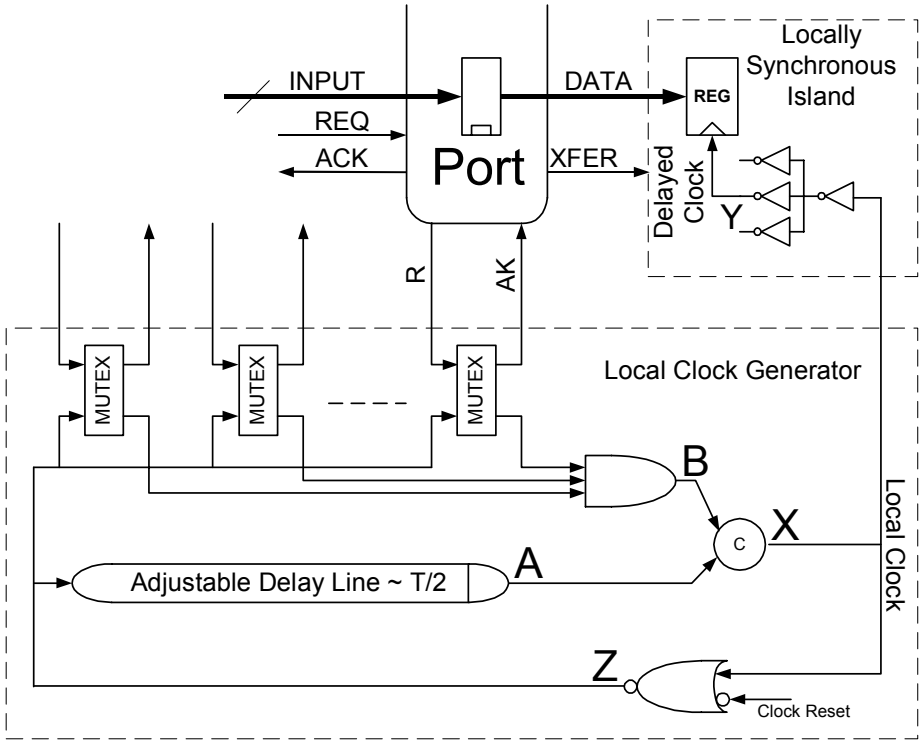

Figure 1: Stoppable Clock Generation [1]

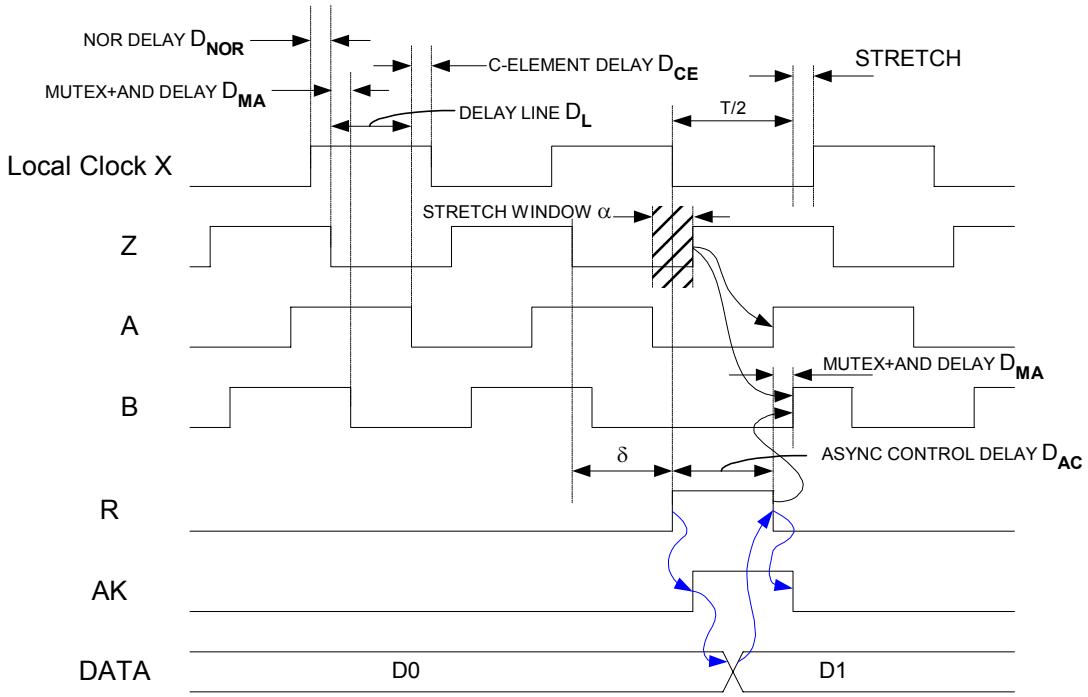

Figure 2: Stoppable Clock Generation – Wave Diagram

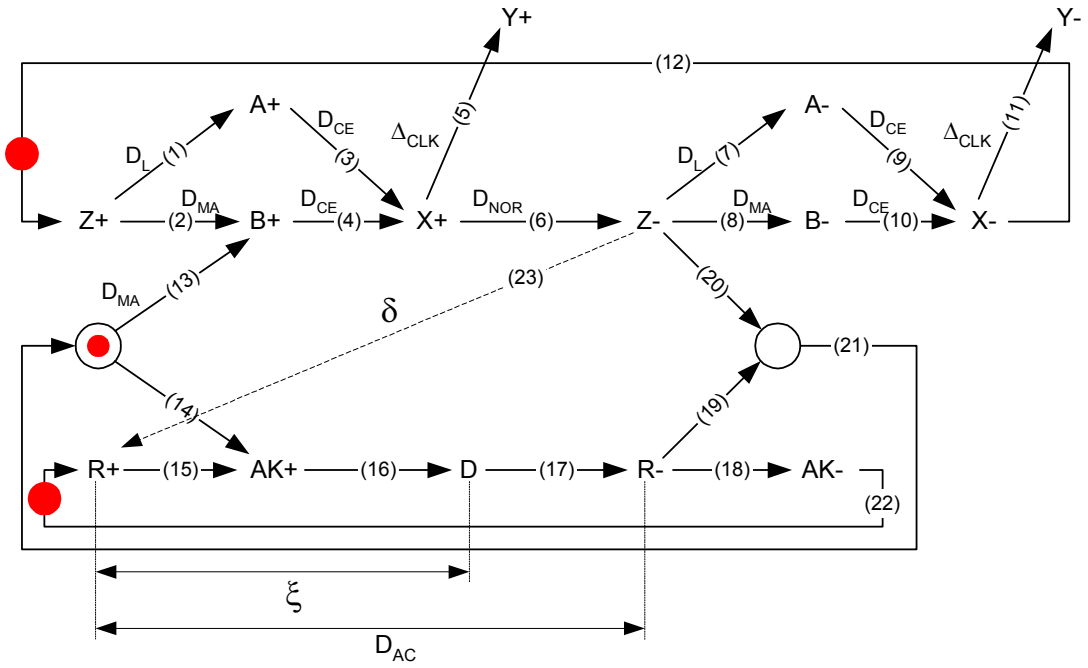

Figure 3: Timed STG of the local stoppable clock of Figure 1

R+ is recognized only when Z=0. Clock cycle stretching occurs when R+ arrives during a *stretch* window, α, towards the end of the low phase of Clock Z. If R+ arrives outside the stretch window, port handshake is over in time (B+ precedes A+), causing no stretch. Once a stretch is started, its maximal length is α. This process can also be described with a timed STG (Figure 3). We add a line tagged δ for the time from Z– to R+, and observe that certain timing of R+ may make the δ arc part of the critical path, stretching the clock, as follows. Let $\delta' \in [0, T)$ be the time between Z+ and R+. Since R+ is ignored when Z=1, we define δ as the effective time between Z– and a port request, as follows:

$$\delta = \begin{cases} \delta' - \dfrac{T}{2}, & \dfrac{T}{2} < \delta' < T \\ 0, & 0 \le \delta' \le \dfrac{T}{2} \end{cases} \tag{1}$$

Note that $0 \le \delta < T/2$. From Figure 3 it can be observed that a stretch occurs if path (6)→(23)→(15)→(16)→(17)→(19)→(21)→(13)→(4) takes longer than a clock cycle:

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

where the various delays are defined in Figure 2. Stated otherwise, the stretch condition is:

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

Subtracting the lower bound from the upper bound, we obtain α, the size of the stretch window:

$$\alpha = (D_{AC} + D_{MA} + D_{CE} + D_{NOR}) - \frac{T}{2} \tag{4}$$

If the circuit's clock cycle is relatively long, then the clock will never be stretched and parameter α of Eq. (4) will assume a negative value. For instance, if each of the constant delays in Eq. (4) were 1 ns, operating at slower than 125MHz will guarantee no clock stretching.

3. Synchronization Failures in GALS Systems

The approach described in Section 2 disregards the delay Δ_{CLK} along the clock tree (from node X to Y), thus potentially causing metastability events in the sampling REG of the Locally Synchronous Island (Figure 1). A failure scenario is depicted in Figure 4.

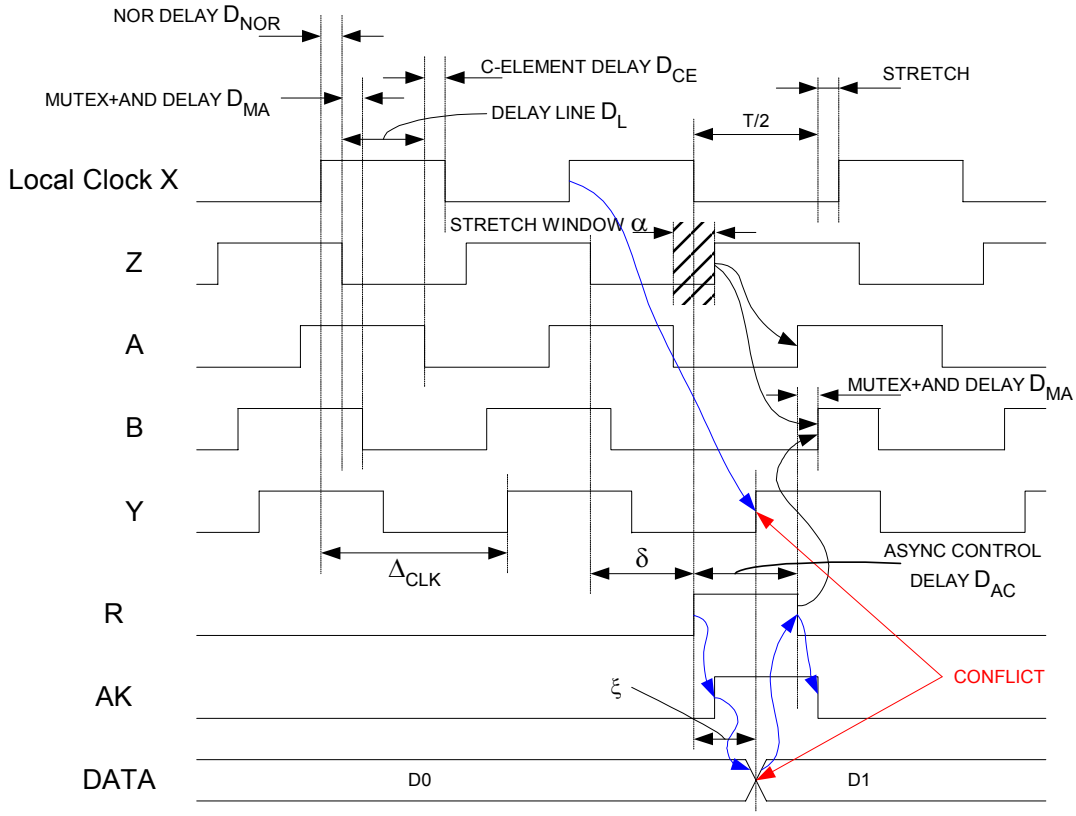

Figure 4: Conflict Example

Let's assume that such a request comes δ after Z– and is granted by the MUTEX. Uncorrelated with the input handshake, the delayed Clock Y may rise simultaneously with the asynchronous data latching in the Port. The conflict may cause metastability in the input REG of the Synchronous Island. Note that, even though Figure 4 presents the conflict during a stretched cycle, the conflict may happen also when no stretch of the clock occurs, since these two events are not correlated.

In addition to the main problem of metastability, this approach suffers from two other drawbacks: pausing the local clock slows down the entire Synchronous Island, and the slowdown may be exacerbated with multi-port GALS modules, where the probability of pausing the clock is higher.

Starting from X+, the conflict occurs when:

$$\Delta_{CLK} = \delta + \xi + D_{NOR} \tag{5}$$

namely, when the delay along arcs (6)→(23)→(15)→(16) matches the delay along arc (5) in Figure 3. The conflict occurs when Y+ happens inside a "danger window" W (setup+hold time) around $\delta + \xi + D_{NOR} + k \cdot T$, where k is an integer (k>0 accounts for clock delays longer than T). The δ statistics is not known, but we believe that the probability of distortion grows with the number of GALS module ports. Figure 5 emphasizes graphically the combinations of Δ_{CLK} and δ that lead to conflicts. Note that for some values of Δ_{CLK}, independent of δ, no conflict can happen (regions S in Figure 5). Alternative solutions that avoid such conflicts are described in Section 4.

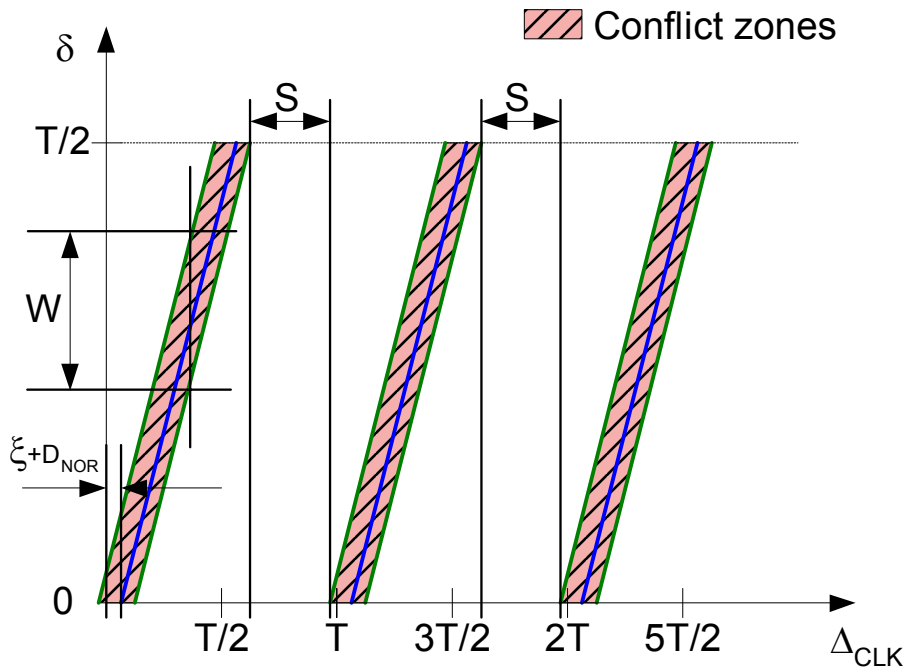

Figure 5: Conflict Zones

Clock tree delays depend on both technology and architecture. Clock tree balancing becomes increasingly difficult for high-performance large SoC designs, incurring higher clock tree delays. For instance, in a 0.18µm technology, a typical clock frequency achievable with standard EDA tools and standard libraries is 100–500MHz (T=2–10ns), while typical clock delays are 1–2ns, depending on module size (some examples are presented in Table 1). Large SoCs, with tens of modules, may require much longer clock delays, approaching T. With faster technologies and larger chips, $\Delta_{CLK} > T$ will be a common case if a single global synchronous clock is attempted for the entire SoC. Thus, while $\delta \in [0, T/2)$, the range of the clock tree delay is not limited by T.

Design	Clock Frequency F	Clock Period T	Clock Skew	Clock Tree Delay
DLX-SYNC (FF)	278 MHz	3.60 ns	60 ps	530 ps (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 Contr. (OC)	200 MHz	5 ns	126 ps	1.014 ns (20% of T)
[2 clocks]	100 MHz	10 ns	137 ps	1.016 ns (10% of T)

Table 1: Clock Tree Delays – Implementation Examples

4. Metastability-Free GALS Clocking

4.1. Limited Delay Clock Tree

When $\Delta_{CLK} < T$, it may be possible to verify that a conflict will never occur. This is performed by timing analysis of the physical design and verifying that:

$$\Delta_{CLK} < D_{NOR} + \xi - T_H \ \cup \ \Delta_{CLK} > D_{NOR} + \frac{T}{2} + \xi + T_{SU} \tag{6}$$

T_{SU} and T_H are the set-up and hold times of the DFF, respectively. When either rule holds, Y+ will occur only inside the first S region (Figure 5 and Figure 6). The upper bound results from the δ=0 case and the lower bound from the δ=T/2 case. Both cases relate to the (6)→(23)→(15)→(16) path in Figure 3. Note that the relation between ξ and the clock cycle varies depending on the clock rate and the technology

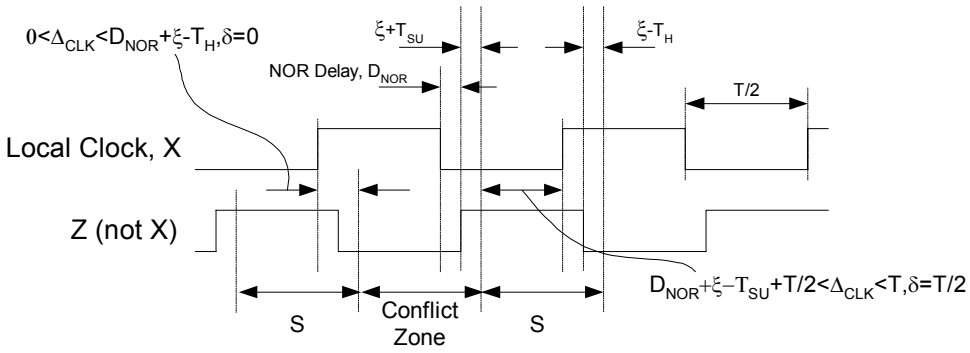

Figure 6: Hazard / No Hazard Windows Example

This solution suffers from a number of disadvantages. First, it must be verified manually after each layout iteration and clock tree design; the solution is not scalable and it may be sensitive to thermal and power supply voltage changes (different changes in ξ, T_{SU}, T_H and D_{NOR}). In addition, ξ is not easy to determine accurately.

4.2. Long Delay Clock Tree

We can generalize the "limited delay clock tree" solution for long delay clock trees having Δ_{CLK}>T as follows. The port access is allowed only during the S intervals (Figure 5 and Figure 6). To prevent metastability, the following inequality must be verified:

$$\begin{aligned} &\Delta_{CLK} < D_{NOR} + \xi - T_H \quad \cup \\ &D_{NOR} + \frac{T}{2} + \xi + T_{SU} + k \cdot T < \Delta_{CLK} < D_{NOR} + \xi - T_H + (k+1) \cdot T \\ &Where \ \ k = 0,1,2,3... \end{aligned} \tag{7}$$

The pros and cons of this approach are similar to the previous one.

4.3. Asynchronous Synchronizer

GALS modules may also avoid clock arbitration and employ standard asynchronous synchronizers (Figure 7). The resolving time of the synchronizer should meet MTBF requirements; in Figure 7 we assume that one clock cycle provides sufficient time for metastability resolution. No clock delay verification is requited, but the interface data rate is affected drastically (data can not be transferred every cycle).

Assuming mesochronous operation (the same clock frequencies at the transmitter and receiver), the minimal data cycle time (REQ+ → REQ+) takes seven clock cycles in the worst case (REQ+ happens immediately following CLK+ and the transmitter and receiver clocks are in phase), as shown in Figure 8. This data cycle can be reduced when the clocks are out of phase (five clock cycles), or by employing a two-phase protocol (down to three clock cycles).

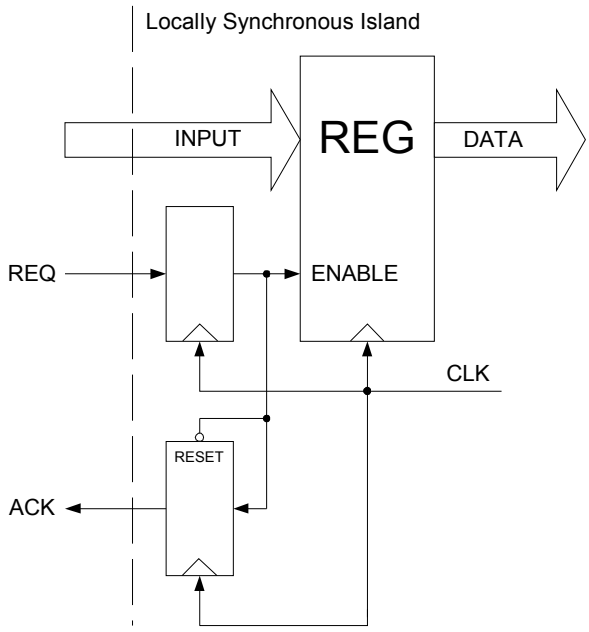

Figure 7: GALS Synchronization with Synchronizer

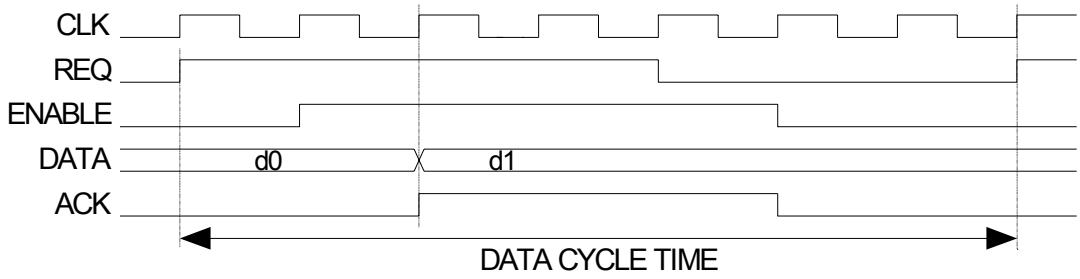

Figure 8: Data Cycle Time

4.4. Matched Delay Port Control

The metastability problem can be solved by inserting delay lines into the circuit of Figure 1, matching the clock-tree delay Δ_{CLK}, as shown in Figure 9. The use of this matched delay may cause longer clock stretching, as demonstrated in Figure 10. 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):

$$\alpha' = \begin{cases} \alpha + \Delta_{CLK}, & \Delta_{CLK} \le \frac{T}{2} - \alpha \\ \frac{T}{2}, & \Delta_{CLK} > \frac{T}{2} - \alpha \end{cases} \tag{8}$$

This is a "slow" architecture: In designs with high clock rates and long clock delays, the clock stretch will happen each handshake, since in this case $\alpha \rightarrow T/2$. In addition, this approach suffers from the same disadvantages as the "limited delay clock tree" above.

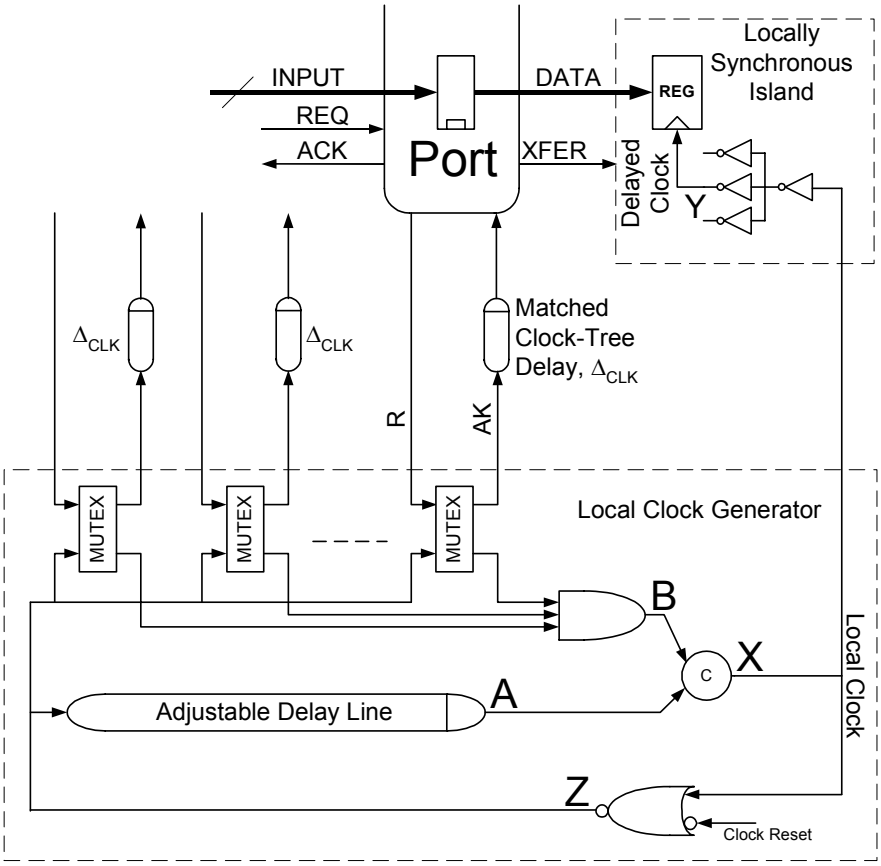

Figure 9: Stoppable Clock Generation with Matched Clock-Tree Delays

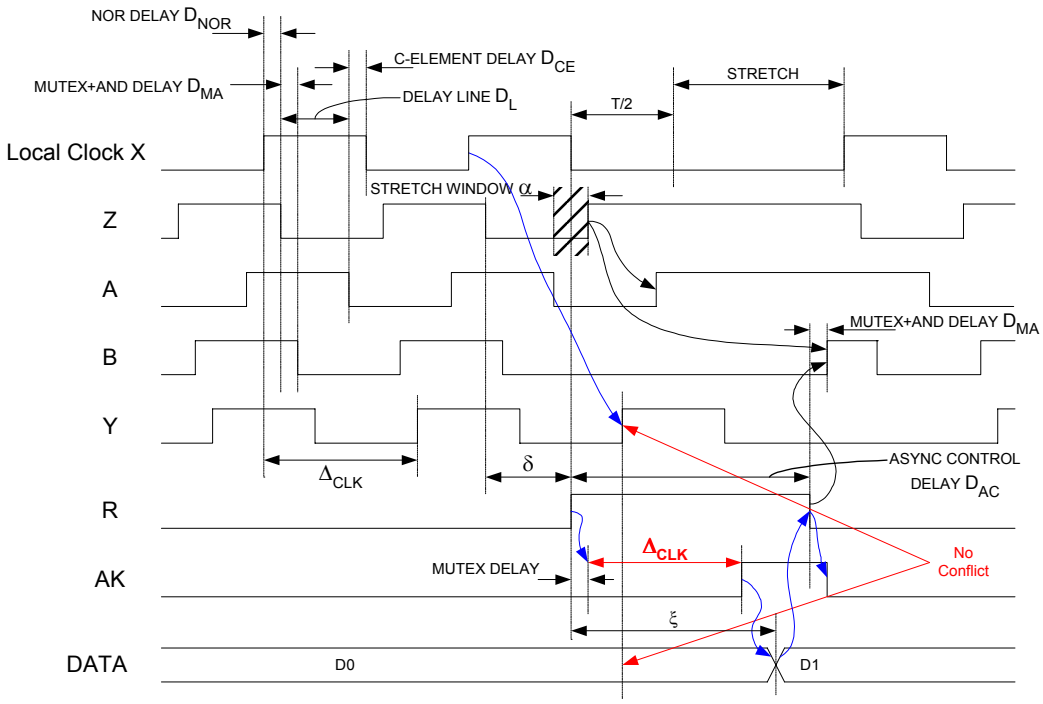

Figure 10: Matched Delay Port Control – Wave Diagram

4.5. Locally Delayed Latching

The following approach eliminates the arbitrated clock (similar to using the asynchronous synchronizer, Section 4.3), and instead synchronizes incoming data by means of locally delayed sampling. The asynchronous controller of the input port (Figure 11) controls both the input latch and Y1, the clock to the first sampling register; Y, the clock to the rest of the module, is uninterrupted. Various modes of operation are demonstrated in Figure 12.

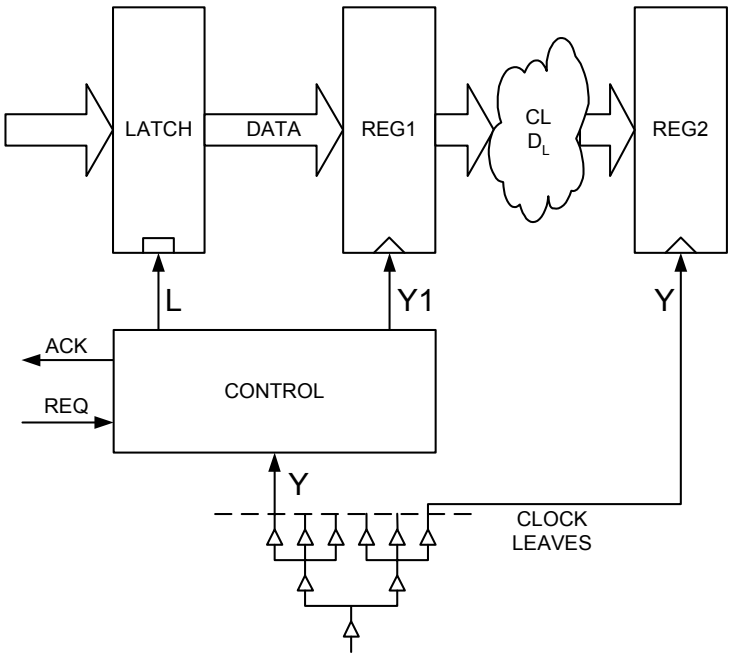

Figure 11: Locally Delayed Latching Conceptual Circuit

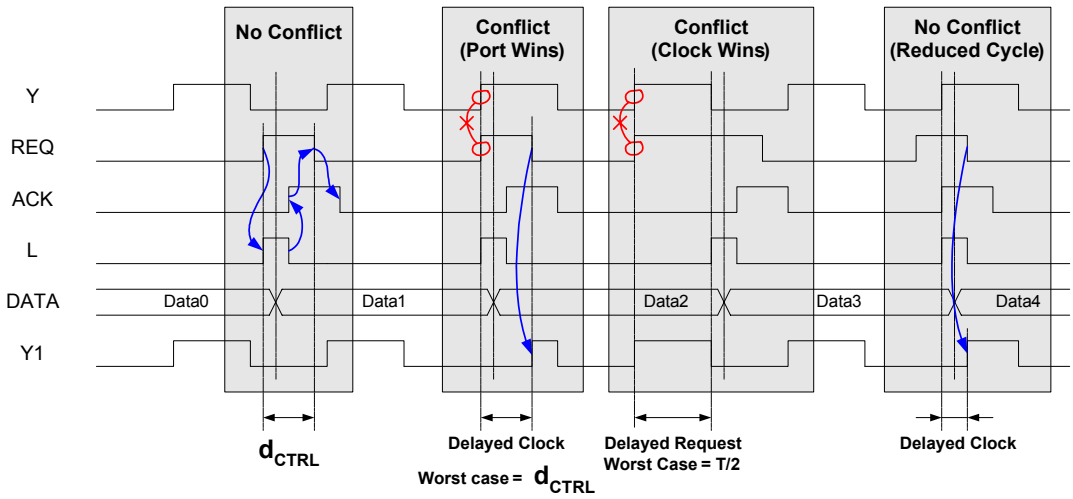

Figure 12: GALS Port Synchronization Technique – Wave Diagram

In this method the clock to the entire 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 treated only during the low-phase of Y, latching the incoming data (L+) and delaying Y1+ when needed. The conflicts between Y+ and REQ+ are resolved by a MUTEX

inside the control. A number of such asynchronous controllers for generating L and Y1 are presented in the following sections.

The main issues in this approach are the latency incurred by the asynchronous control, D_{CTRL}, and the MUTEX resolution latency. We define a minimally allowed high-phase time T_{HP}^{Min} (typically about three FO4 inverter gate delays). In addition, we define the maximal time that the MUTEX require to resolve metastability, $T_{Metastab}^{MUTEX}$. Then we require that

$$\begin{aligned} &\frac{T}{2} - D_{CTRL} - T_{Metastab}^{MUTEX} > T_{HP}^{Min} \\ &\Leftrightarrow \\ &D_{CTRL} < \frac{T}{2} - T_{HP}^{Min} - T_{Metastab}^{MUTEX} \end{aligned} \tag{9}$$

In order to prevent metastability in REG2, the following should be satisfied (D_L is the latency of the combinational logic between REG1 and REG2):

$$\begin{aligned} &D_L + T_H < T - D_{CTRL} - T_{Metastab}^{MUTEX} \\ &\Leftrightarrow \\ &D_L < T - T_H - D_{CTRL} - T_{Metastab}^{MUTEX} \end{aligned} \tag{10}$$

The MUTEX metastability can be tolerated if the clock period is long enough to allow for the resolution of any metastability as well as propagation through the logic that lies in the path to the next register.

Using a standard formula for MTBF [9] and operating at 200MHz with 0.13μm technology (τ=30ps and W=60ps), preserving one quarter of the clock cycle for MUTEX resolution, we obtain MTBF of about 3,000 years. With a 0.35μm technology (τ=100ps, W=200ps and F_C=65MHz, as used for the simulations in Section 5), the MTBF grows to 10,000 years. For faster operation rate (e.g. 400 MHz at 0.13μm technology) more complicated circuits are required, preserving up to T/2 for MUTEX resolution rather than T/4.

Preserving one quarter of the clock cycle for MUTEX resolution, Eqs. (9), (10) are updated as follows:

$$\begin{aligned} &D_{CTRL} < \frac{T}{2} - T_{HP}^{Min} - T_{Metastab}^{MUTEX} \quad \text{where } T_{Metastab}^{MUTEX} = T/4 \\ \Rightarrow \quad &D_{CTRL} < \frac{T}{4} - T_{HP}^{Min} \end{aligned} \tag{11}$$

And the logic delay requirement, D_L, is modified as follows:

$$\begin{aligned} &D_L < T - T_H - D_{CTRL} - T_{Metastab}^{MUTEX} \quad \text{where } T_{Metastab}^{MUTEX} = T/4 \\ \Rightarrow &D_L < \frac{3 \cdot T}{4} - T_H - D_{CTRL} \end{aligned} \tag{12}$$

D_{CTRL} contains additional buffering delays when wide data path is required. These constraints are verified in Section 5 for all of the following implementations.

4.5.1. Decoupled Input Port

Figure 13 shows an implementation of Figure 11. Without a conflict, Y1+ is either not delayed or delayed by less than D_{CTRL}. R2+ is granted only during the low-phase 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→Di to open the latch and then close it again (L+→L–). After R2–, Y1+ triggers REG1, leading to a shortened cycle in the combinational logic following REG1 (the cycle is shortened by D_{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 Figure 14.

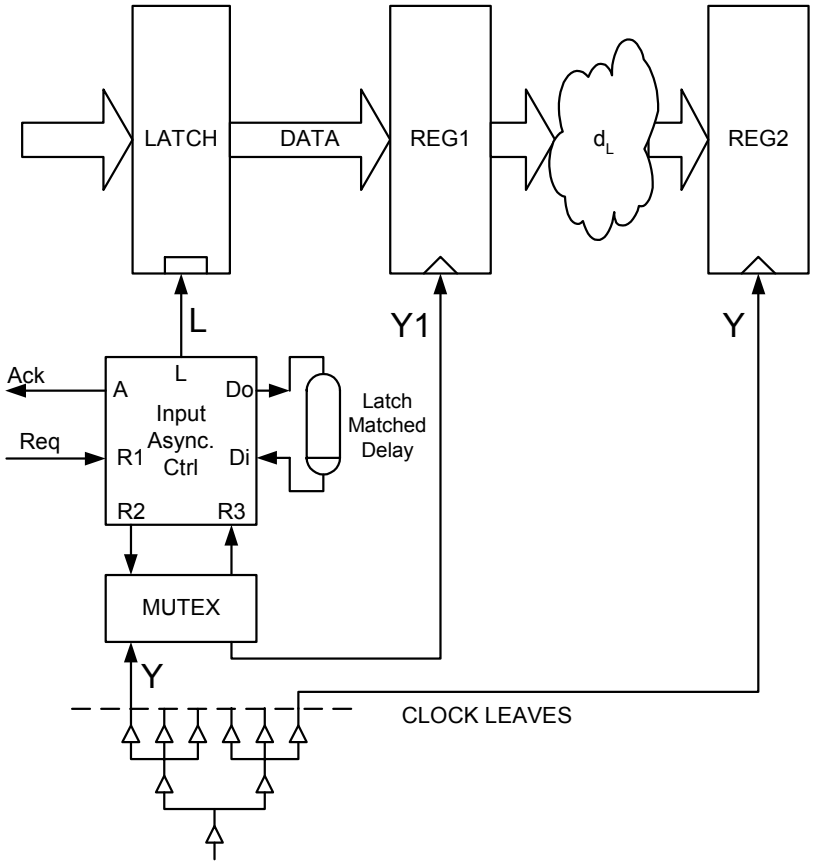

Figure 13: GALS Module Decoupled Input Port

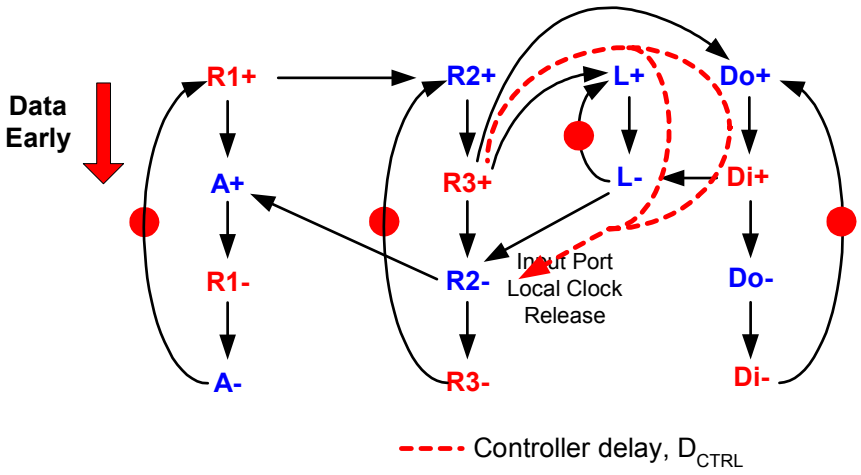

Figure 14: GALS Decoupled Input Port Asynchronous Control STG

The controller delay is measured along the red-dashed path. The path is contained entirely inside the input port of the synchronous island, thus we ensure that the clock cycle reduction depends solely on

the input port control logic (and it does not depend on the logic and clock of the transmitter module). The MUTEX output Y1 should be buffered 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 5.

4.5.2. Decoupled Output Port

The transmitter circuit is shown in Figure 15. The internal acknowledge (A1) is decoupled from the external asynchronous handshake.

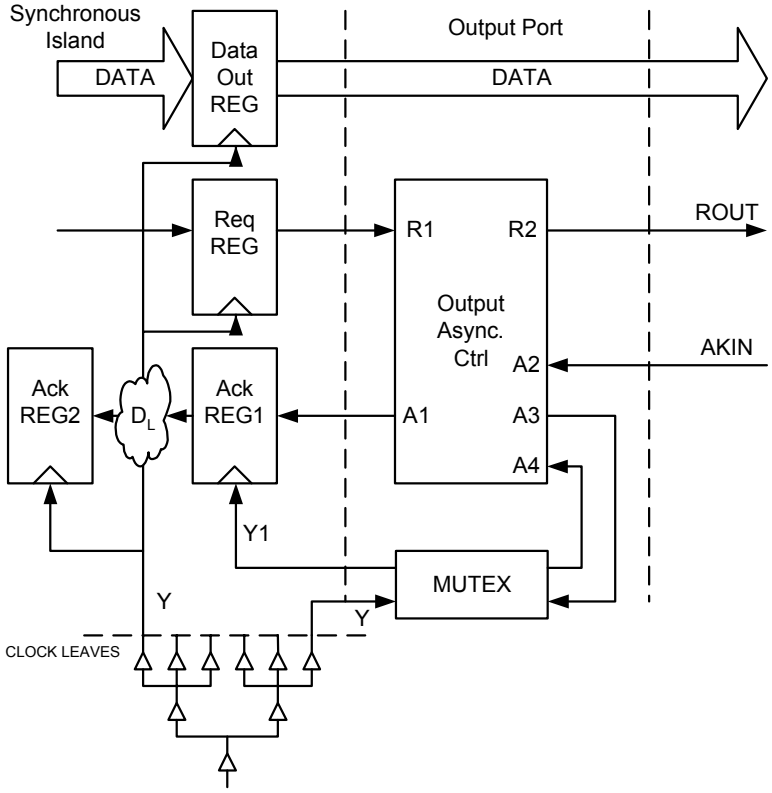

Figure 15: GALS Module Decoupled Output Port

At the beginning, the synchronous island posts the data and initiates the internal request (R1+), which is passed directly to the external interface (R2+). When the acknowledgement is received (A2+), it is passed to the MUTEX (A3+) and in parallel the external request R2 is released (R2–). The MUTEX resolves any conflict between A3+ and the local clock Y. When A4+ wins over Y+, A1 is set. The design assumes that the controller delay [R1– → A1–] is much less than the clock period, which is very realistic. The STG of the Decoupled Output Port control is shown in Figure 16.

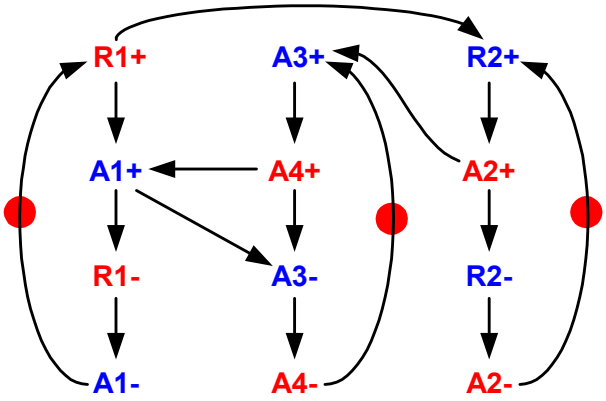

Figure 16: GALS Decoupled Output Port Asynchronous Control STG

The controller latency of the Decoupled Output Port control, d_{CTRL}^{OUT}, should be verified according to Eqs. (11) and (12). In this case:

$$d_{CTRL}^{OUT} = D[A4+ \rightarrow A1+ \rightarrow A3-] \tag{13}$$

The latency of the controller is verified in Section 5.

4.5.3. A Simpler Input Port

The following architecture simplifies the Receiver side, eliminating the asynchronous controller from the Input Port.

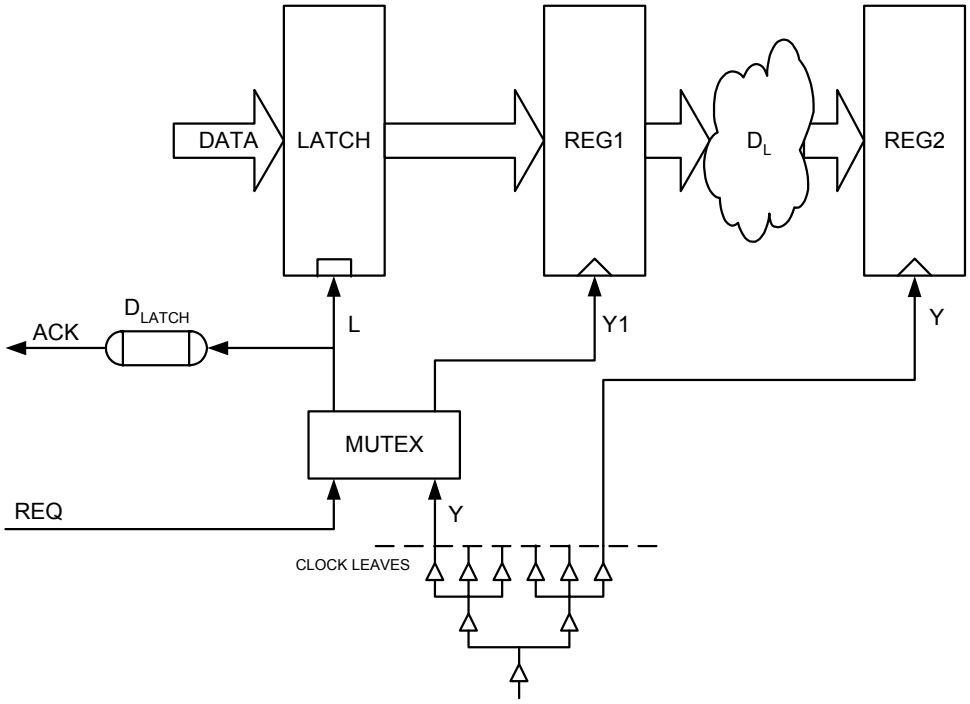

Figure 17: Simple Input Port

The receiver delay D_{CTRL}, now depends on the external delays of the transmitter:

$$D_{CTRL} = D_{LATCH} + D_{TRANSMITTER}(\text{ACK}+ \rightarrow \text{REQ}-) \quad (14)$$

The Latch Matched Delay (Figure 17) could have been is reduced from D_{LATCH} value to D_{LATCH}-$D_{TRANSMITTER}$, but $D_{TRANSMITTER}$ is unknown *a-priori*, and therefore it is better to set the Latch Matched Delay to D_{LATCH}. This simple input port is compatible with the output port of Section 4.5.2. The latency of this constellation (Simple Input Port with the Decoupled Output Port) is also verified in Section 5.

5. Simulations

The circuits of Section 4.5 were synthesized using Petrify, converted to VHDL, synthesized by the Synopsis Design Compiler using 0.35μm CMOS libraries, and verified by gate level simulations with wire-load model delays (SDF). Table 2 lists the results for the three controllers.

These results are based on data bus width of 16 bits. In general, we assume a clock cycle of 160 FO4 inverter delays in standard ASIC [10]. One quarter cycle is preserved for metastability resolution and another quarter cycle (40 inverter delays) are dedicated to the delay of the asynchronous controller and for high-phase generation. Let's assume that the controller consumes about 25 inverter delays out of the said 40 in the second quarter of the cycle. This leaves 15 gate delays for generating the high phase of the clock. Our 0.35μm library specifies $\boldsymbol{T_{HP}^{Min}} = 0.361\boldsymbol{ns}$, namely about 3 inverter delays. Thus,

$$\boldsymbol{T_{HP}^{Min}} < \frac{\boldsymbol{T}}{4} - \boldsymbol{d_{Ctrl}}$$

$$\Leftrightarrow$$

$$0.361\boldsymbol{ns} < 1.8\boldsymbol{ns} = 0.12\boldsymbol{ns} \times 15$$

as required by Eq. (11).

Circuit	Critical Path	Latency (0.35μm)	Num. of FO4 inverter delays
Decoupled Input Port	R3+→Do+→Di+→L–→R2–	3.132 ns	24
Decoupled Output Port	A4+→A1+→A3–	1.811 ns	13
Simple Input Port with Decoupled Output Port	Latch Delay→A2+→R2–	2.139 ns	14

Table 2: Controllers Delays

6. Conclusions

Previously proposed locally generated, arbitrated clocks for GALS SoCs [1][5][6][7] face the risk of synchronization failures if clock delays are not accounted for. The problem has been analyzed based on clock delays, cycle time, and complexity of the asynchronous port controllers. A few methods have been presented. In some cases, it is sufficient to extract all the delays and verify whether the system is susceptible to metastability. In other cases, when high data bandwidth is not required, asynchronous synchronizers or matched-delay asynchronous ports may be employed. Arbitrated clocks may be traded off for locally delayed input and output ports, facilitating high data rates. The latter circuits have been simulated, to verify their performance.

References

[1] Simon Moore, George Taylor, Robert Mullins, Peter Robinson, "Point to Point GALS Interconnect", Proc. Of ASYNC'02, Manchester, UK, April 2002.

[2] E. G. Friedman, "Clock Distribution Networks in Synchronous Digital Integrated Circuits," Proc. of the IEEE, vol. 89, pp. 665-692, 2001.

[3] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems," Stanford University, 1984.

[4] J. Kessels, A. Peeters, P. Wielage, and S.-J. Kim, "Clock Synchronization through Handshake Signalling," in Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, 2002, pp. 59--68.

[5] Stephan Oetiker, Frank K. Gürkaynak, Thomas Villiger, Hubert Kaeslin, Norbert Felber, Wolfgang Fichtner, "Design Flow for a 3-Million Transistor GALS Test Chip," Integrated Systems Laboratory, ETH Zurich, ACiD 27, January 2003.

[6] Thomas Villiger, Hubert Kaeslin, Frank K. Gürkaynak, Stephan Oetiker, Wolfgang Fichtner, "Self-Timed Ring for Globally-Asynchronous Locally-Synchronous Systems," Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC'03), pp. 141-150.

[7] Jens Muttersbach, Thomas Villiger, and Wolfgang Fichtner, "Practical Design of Globally-Asynchrounous Locally-Synchronous Systems," 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems, Eilat, Israel, April 2000, pp. 52-61.

[8] 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. International Conf. Computer Design (ICCD), 2000.

[9] C. Dike and E. Burton, "Miller and Noise Effects in a Synchronizing Flip-flop," IEEE Journal of Solid-State Circuits, 34(6), pp. 849-855, 1999.

[10] International Technology Roadmap for Semiconductors (ITRS), 2001.