International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 1, Issue 5, May2014)

LOW AREA MAXIMAL THROUGHPUT COMMUNICATION INTERFACE FOR GLOBALLY RATIOCHRONOUS LOCALLY SYNCHRONOUS GRLS SYSTEM

Naveen Kumar D B
M. Tech, Dept. of ECE, TOCE, Bangalore

Mrs. K V Meena
Asst. Prof, Dept. of ECE, TOCE, Bangalore

Abstract— **This paper represent a source-synchronousadaptive interface for the globally ratiochronous, locally synchronousdesign style, a subset of the globally asynchronous, locally synchronous (GALS) design style in which the frequencies of all clocks are not phase-aligned but are constrained to be rationally related, i.e., they are all submultiple of the same physical or virtual frequency. The interface can be designed using only standard cells and guarantees maximal throughput in addition to an average latency four times lower compared with state-of-the-art asynchronous first-input, first-output GALSinterfaces.**

Index Terms—**Application specific integrated circuits, asynchronous circuits, circuits and systems, system-on-a-chip.**

I. INTRODUCTION

Up to the mid-1990s, the globally synchronous designstyle was seen as the best choice for electronic systems design it is simple, reliable, and supported by many wellestablished design tools. Since then, however, technology scaling has considerably worsened the problems and limitations of globally synchronous systems, and the VLSI design industry has started looking for alternatives. According to the historical perspective in this surge of interest in nonsynchronous design styles started happening in the late 1990s, from the 250-nm technology node. Nowadays, most systems include multiple clock domains. Several different needs are driving the research in novel clocking and synchronization methods for complex systems-on-chip (SoCs).

1) In terms of engineering effort, silicon and power, it is expensive to maintain the globally syncshronous assumption since the number of clock tree leaves roughly doubles with every new technology node .

2) The buy and assemble model of building SoCs is making systems increasingly modular. Hierarchical physical design is required to avoid costly timing closure iterations for every small change in any part of the design; such iterations drive up the NRE costs of SoCsand impede and discourage design space exploration. Hierarchical physical design can be guaranteed with a truly latency-insensitive design style.

3) Variations due to manufacturing and operating conditions require adaptive synchronization techniques.

4) Power management techniques like per-module dynamic voltage frequency scaling (DVFS) are necessary to guarantee low-power operation. Their deployment demands the ability to safely cross clock domain boundaries between clocks running at different clock frequencies.

5) The interfaces must have limited overheads and must be easily integrated in the standard EDA design flow.

6) The interfaces must guarantee maximal throughput and low latency, to ensure high performances. This is particularly Important for systems in which latency determines throughput, such as Networks-on-Chip.Some argue that the industry should move to a fully asynchronous design style, which would have the advantage to completely eliminate the timing closure problem. However, there is a lack of established and reliable asynchronousdesign tools and the synchronous IP libraries accumulated in the past decades would have to be completely redesigned if an asynchronous design style was adopted. Although the globally synchronous assumption is hard and expensive to maintain across the whole chip, the synchronous design style is still the best style for module-level design, because modules have normally only limited size. This has lead to the recent success of globally nonsynchronous (GnS) design styles, a series of design styles that take the best from both the synchronous and the asynchronous worlds. The individual modules remain synchronous but they all run at their own clock. The different modules communicate using special clock-domain-crossing techniques. At module-level, a synchronous design style is used, so that the module-level Design flow can be based on the standard and well-established synchronous design flow. At chip-level, no global balanced clock tree is present, and so global timing closure is not a problem. There are two main flavors of globally nonsynchronous design styles: the mesochronous design style and the globally asynchronous, locally synchronous (GALS) design style. In a mesochronous system, the different modules all run at the same frequency, but there is no global balanced clock tree, i.e., the clocks of the different modules are not aligned in phase. Adaptive, latency-insensitive low latency and maximal-throughput interfaces for mesochronoussystems can be designed; thus, the mesochronous design styleis well suited for high-performance systems, supports adaptive hierarchical physical design but does not support DVFS. In a GALS system, the clocks are generated locally and the modules can run at different frequencies. Compared with the

mesochronous design style, GALS supports DVFS. On the other hand, latency-insensitive, adaptive communication interfaces for GALS systems are more complex than communication interfaces for mesochronous systems, and introduce a much higher performance overhead which is acceptable only for some applications.

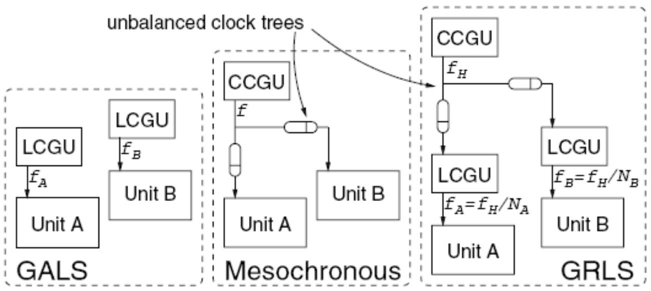


Fig. 1. GALS, mesochronous, and GRLS conceptual diagrams. CCGU: central clock generation unit. LCGU: local clock generation unit.

In general, both mesochronous and GALS design styles have advantages and disadvantages with regards to the needs of present-day SoCs, and most complex systems nowadays contain elements taken from both. We propose a novel GnS design style called the GRLS, which is intermediate between the two approaches. As for GALS, the clock frequencies of the different modules are not necessarily equal. However, compared with GALS, they are constrained to be rationally related, i.e., they are all submultiple of a physical or virtual frequency *fH*. Thus, GRLS supports quantized DVFS. Conceptual block diagrams of the three design styles are shown in Fig. 1. This paper aims at proving that, for a certain class of systems, the GRLS design style is better than the GALS and the mesochronous design styles in satisfying the needs of present-day SoCs. The main topic of this paper is the GRLS interface, for a DVFS efficiency analysis of GRLS we refer the reader to. The main contribution of the paper is to show how the periodic properties of ratiochronous systems allow the definition of adaptive, standard-cells-only and maximal performanceinterfaces having a low area overhead, where data is transferred at the rate of one data item per clock cycle of the slowest of the two communication units. The GRLS interface is source-synchronous, i.e., a synchronization signal traveling from the transmitter to the receiver encodes Transmitter clock information. GRLS interfaces were presented in however; the interface presented in this paper is based on a new concept, i.e., it uses a single clock at the receiver end of the channel and inserts a single delay line on the strobe path instead of two delay lines on the receiver clock. Comparedwith the interface presented in, the area overhead and complexity of the system are reduced while the tolerance to no idealities is increased. Also, this paper presents formal proofs for the properties of the GRLS interface. The main advantages of the GRLS interface can be summarized as follows.

1) Globally nonsynchronous design style supporting hierarchical physical design.

2) Based on a continuous learning phase, the interface is Adaptive and can cope with no idealities such as clock jitters and propagation delay misalignments.

3) Unlike mesochronous interfaces, GRLS interfaces support quantized per-module DVFS. Quantization has only a limited impact on DVFS efficiency compared with GALS.

4) Low area overhead comparable with GALS and mesochronous interfaces; except for a delay line, the interface is a synthesizable RTL design. The delay line is fully digital but is designed at gate level using standard cells.

5) The interface guarantees maximal throughput and has a much lower latency overhead compared with GALS interfaces. Latency figures are essentially the same as for the fastest mesochronous interfaces that can be found in Literature.

The main limitation of the GRLS design style is that it is a viable solution only when the least common multiple *fH*between the clock frequencies of the transmitter and theReceiver is below a certain upper bound, which can be calculated as a function of the no idealities of the system such as clock jitters, propagation delay misalignments, and so on. For a 90-nm technology scenario, an upper bound of *fH*<1 GHz will be calculated in this paper. Thus, GRLS is viable for systems communicating at relatively low frequency and/or for systems in which the ratio of the transmitter and the receiver clock frequencies presents small integers at both the numerator and the denominator.

The remainder of the paper is organized as follows. In Section II, the GRLS communication interface is introduced and analyzed. In Section III, having completed the presentation of the GRLS interface, other state-of-the-art communication interfaces used in nonsynchronous communication scenarios are reviewed. In Section IV, the standard-cells implementation of the GRLS interface is presented. In Sections V and VI, respectively, area overhead and robustness of the interface—two industry needs identified in this section—are analyzed. Section VII concludes the analysis of the needs by rigorously comparing the performances of the interface with state-of the art GALS and mesochronous interfaces. Section VIII concludes the paper. In Appendix A, are reported the formal proofs of different interface properties that are used in this paper.

II. GRLS INTERFACE

In a GRLS system all local clock frequencies aresubmultiple of a physical or virtual frequency *fH*and are all rationally-related. The GRLS interface is the component allowing different GRLS modules to communicate together.

A. Communication Problem Formulation and Notation

The GRLS communication problem consists in interfacing a synchronous transmitter module with a synchronous receiver module. The two units are clocked, respectively, by the two clocks *clkT*and *clk R*, running, respectively, at frequencies *fT*and *fR*(the subscripts *T* and *R* indicate the transmitter andthe receiver, respectively), with

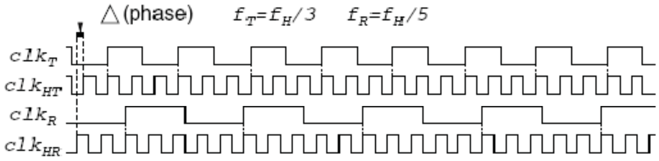


Fig. 2.Definition of phase difference in a GRLS system.

fT=1/*TT* =*Fh*/*NT* ;
fR=/1*TR* =*fH*/*NR*.

Both clocks are submultiple of a virtual or physical frequency
fH=1/*TH* = *NT***fT*= *NR***fR*.

In a multifrequency implementation with global frequencies { *fH*1, *fH*2, . . .}, *fH*is a virtual frequency, defined as the least common multiple between the global clock frequencies, i.e., *fH*= *lcm* (*fH*1, *fH*2, . . .). The definition of skew, or phase difference, Used formesochronousa system does not support clocks running at different frequencies and must be extended. Two additional virtual clocks *clkHT*and *clkHR*are defined. Both clocks run at frequency *fH*. The edges of *clkHT*are synchronous to the edges of *clkT*and the edges of *clkHR*are synchronous to the edges of *clk R*, as shown in Fig. 2. The skew __ between *clkT*and *clkR*is defined as the skew between*clkHT*and *clkHR*. The unidirectional GRLS communication problem is then defined as follows: to synchronize data between the *clkT*and the *clk R* clock domains, running at rationally related frequencies with an unknown skew between the clocks. The GRLS communication problem is a subset of the GALS communication problem, where *clkT*and *clk R* are not constrained to be clocked at rationally related frequencies; it is also a superset of the mesochronous communication problem, in which *fT*= *fR*= *f* .

B. Key Insight

In all GALS interfaces that have so far been proposed when the transmitter has a new data item to transmit, it first informs the receiver about the upcoming transmission and does not begin transmission until the receiver has not been informed and has prepared itself to receive data. This solution is necessary to achieve synchronization when no assumption is made on the two clocks. However, it also carries an intrinsic latency penalty. Mesochronous interfaces such as are based on a completely different concept compared with GALS interfaces (see Section III). The receiver, knowing that the transmitter and receiver clocks run at the same frequency, performs a learning phase and understands on which time instants data can be safely read. When the transmitter has data to transmit, it does not need to inform the receiver, and instead outputs data immediately and is sure that the receiver will read it as soonas the data item can be safely sampled.

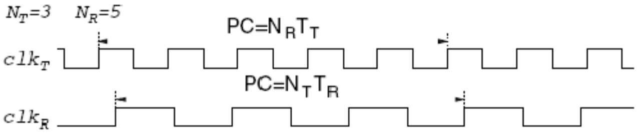


Fig 3 Periodicity cycle definition in GRLS system

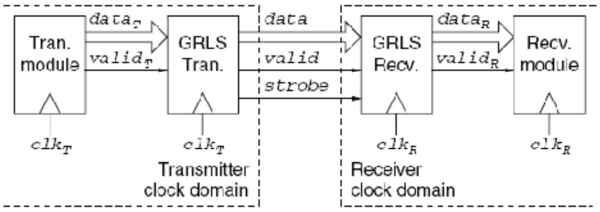


Fig 4 Block Diagram of a GRLS interface

Drastically reduced latency figures are obtained. In principle, it is impossible to use a learning-phase solution to solve the GALS communication problem, because it is not possible to predict when the clock edges of the two clocks will take place. However, the periodic properties of rationally related systems allow the design of learning-phase interfaces for GRLS systems. A source-synchronous strobe signal is used to perform the learning phase.

C. Overview

In a GRLS link, the alignment between clock edges is periodic with period *PC* = *NRTT* = *NT TR* (see Fig. 3), PC is periodicity cycle. Because of this property, the GRLS communication problem is inherently simpler than the GALS communication problem. The basic block diagram of a GRLS interface is shown in Fig. 4. To synchronize data between the transmitter and the receiver modules, a GRLS transmitter and a GRLS receiver are introduced between the two units. The GRLS transmitters synchronous to the transmitter module while the GRLS receiver is synchronous to the receiver module. Data items originate in the transmitter module, which sends them to the GRLS transmitter using the *data T*lines. The *validT* line is set to zero when the transmitter module has no valid data item to output. Valid data items are stored in a first input, first-output (FIFO) buffer in the GRLS transmitter until they can be output. The minimal depth of the FIFO buffer depends on the characteristics of the system (see Section II-D).

The GRLS transmitter outputs data on a subset of its rising clock edges, called output edges. On no output edges, the data lines are kept stable. On every output edge, data is output; if the GRLS transmitter has nothing to output, it outputs a dummy data item which is marked as such by ideas setting the *valid* line, an additional data line introduced to distinguish valid and dummy data items. A regulation algorithm, which knows the values of *NT* and *NR* and is presented in Section II-D, establishes which edges of the transmitter clock are output edges. One property of the regulation algorithm is that the output edges are periodic with period PC = *NT NRTH*, i.e., if a data item is output at time *t*, another data item is output

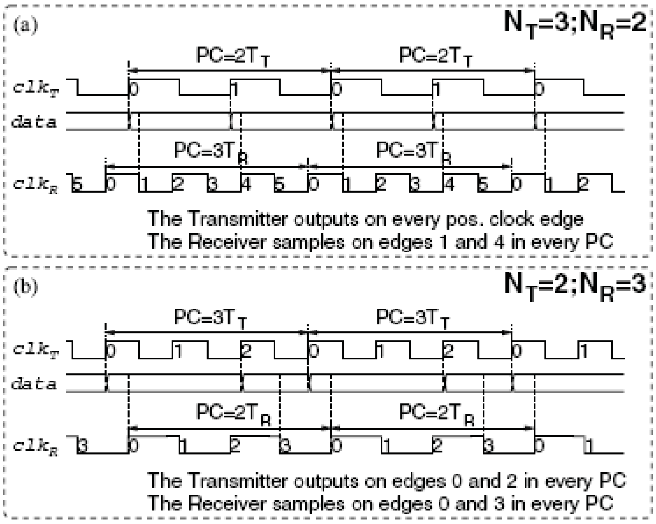

(a)
N_T=3;N_R=2
PC=2T_T
PC=2T_T
clk_T
data
PC=3T_R
PC=3T_R
clk_R
The Transmitter outputs on every pos. clock edge
The Receiver samples on edges 1 and 4 in every PC
(b)
N_T=2;N_R=3
PC=3T_T
PC=3T_T
clk_T
data
PC=2T_R
PC=2T_R
clk_R
The Transmitter outputs on edges 0 and 2 in every PC
The Receiver samples on edges 0 and 3 in every PC

Fig. 5. Periodic output and sampling patterns in a GRLS interface. (a) Fast receiver(*fR>fT*). (b) Fast-transmitter (*fT>fR*)

at time *t* + *PC*. If the receiver runs faster than the transmitter, the GRLS transmitter outputs one data item per clock cycle (the regulation algorithm marks all rising clock edges as output edges). Otherwise, the regulation algorithm selects the output edges so that the GRLS transmitter outputs on average one data item per receiver cycle. Additionally, the output edges are so that every data item is guaranteed to remain stable on the channel for more than half a receiver clock period. Formal proofs of the regulation algorithm properties are given in Appendix A. The GRLS receiver can sample data on both the positive and negative edges of the receiver clock. Since data items are guaranteed to remain stable on the channel for more than half a receiver clock cycle, every data item can in principle be safely sampled at least on a positive or a negative receiver clock edge (by safely sampled we mean that the data lines do not toggle from a setup time before the clock edge to a hold time after the clock edge). Fig. 5 shows relevant signals fora fast-receiver and a fast-transmitter links. In the fast receiver case, data items reach the GRLS receiver at a constant rate (one per transmitter cycle); in the fast-transmitter case, they arrive at a variable rate. The clock edges alignment between transmitter and receiver clocks is periodic with period PC; also, data output times are periodic with period PC due to the periodic nature of the regulation algorithm. Therefore, the alignment between data arrival times at the GRLS receiver and receiver clock edges is also periodic with period PC: if the GRLS receiver can safely sample a data item at one time instant it can safely sample another data item 1 PC later, 2 PC later, 3 PC later, etc. In Fig. 5(a), as an example, the receiver can sample data on edges 1 and 4 in every periodicity cycle; in Fig. 5(b), it can sample data on edges 0 and 3 in every periodicity cycle: these sampling patterns allow every data item to be sampled safely and only once (note that the clock edge on which a periodicity cycle is assumed to start is arbitrary). There may be more than one safe sampling pattern: as an example, in Fig. 5(b), the GRLS receiver could safelysample data on clock edges 1 and 3 instead of edges 0 and 3.

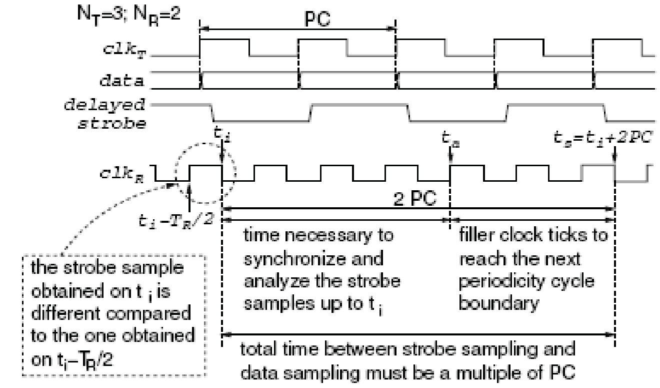

N_T=3; N_R=2
PC
clk_T
data
delayed strobe
t_i
t_a
t_s=t_i+2PC
clk_R
2 PC
t_i−T_R/2
the strobe sample obtained on t_i is different compared to the one obtained on t_i−T_R/2
time necessary to synchronize and analyze the strobe samples up to t_i
filler clock ticks to reach the next periodicity cycle boundary
total time between strobe sampling and data sampling must be a multiple of PC

Fig. 6. Sampling mechanism in a GRLS receiver

The GRLS receiver decides on which edges data should be sampled using a strobe-based source-synchronous mechanism, which is explained in detail in Section II-E. A strobe generated by the GRLS transmitter toggles between 0 and 1 every time a new data item is output by the unit, as shown in Fig. 6. The strobe line is routed bundled together with the data lines, so that ideally all propagation delays through the channel are identical. Section VI analyzes the impact of propagation delay and other non-idealities on the operation of the interface. In the GRLS receiver, the strobe is delayed for a fraction of clock cycle and then continuously sampled on all receiver clock edges. The strobe samples are synchronized to the receiver clock domain using high-latency multistage synchronizers and then analyzed: if the strobe sample obtained on one clock edge, at *ti*, is different compared with the strobe sample obtained half a cycle earlier, then the GRLS receiver deduces that a new data item could have safely been sampled at time *ti*. We show in Section II-E that as long as certain conditions are satisfied, this conclusion is always correct, even when the strobe sampler encountered metastability at *ti*. Due to the synchronization latency, the analysis of the strobe samples obtained at time *ti*takes several clock cycles and is not completed until a time instant *ta >ti*. If, when the analysis is completed, the GRLS receiver determines that a new data item could have been safely sampled at time instant *ti*, it samples data at time instant *ts*= *ti*+ *K PC*, i.e., the first time instant after *ta* falling an integer number of PC after *ti*: the periodic properties that we mentioned guarantee that a new data item can be safely sampled on *ts*. The mechanism is shown in Fig. 6 (*K* PC = 2 PC). The GRLS receiver sampling mechanism is applied continuously and we prove in Section IIE that this guarantees that all data items are safely sampled and no data item is sampled twice. Summarizing, data sampling is determined by the analysis of strobe samples obtained several cycles earlier during a learning phase. Data sampling and learning phase happen continuously and in parallel, i.e., at any time the GRLS receiver is doing in parallel two tasks: 1) sampling data based on a learning phase that took place some cycles earlier and 2) performing a new learning phase that will guide data sampling some cycles later. Should there be any drift in the skew between the transmitter and the receiver clocks, the GRLS receiver continuously updates the sampling pattern to compensate. This is elaborated later in Section VI.

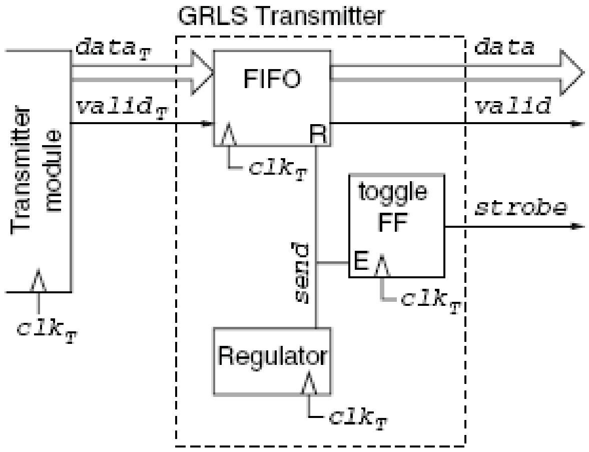
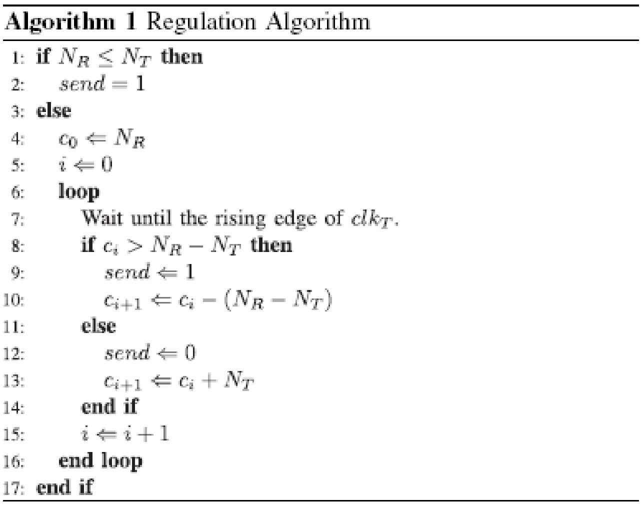


Fig. 7.GRLS transmitter structure.

The GRLS receiver may need to sample more than one data item per clock cycle. In this case, the data item that cannot be consumed immediately by the receiver module must be stored in a FIFO. We prove in Section II-E that a FIFO buffer with depth 1 is sufficient and never overflows. The data items are then passed from the GRLS receiver to the receiver module using the *data R*lines. If the GRLS receiver contains no *valid* data item, it desserts the *valid R* additional data line. The main benefit of the GRLS interface is low latency: although the learning phase takes several cycles to complete, this delay does not impact data latency. When a data item reaches the GRLS receiver, the GRLS receiver is already prepared to receive it and samples it on the first available occasion.

D. GRLS Transmitter

GRLS transmitter is a synchronous block and is clocked with the same clock *clkT*as the transmitter module. The throughput of any data communication link where the transmitter and the receiver operate The GRLS transmitter structure is shown in Fig. 7. That different frequencies is limited to one data item per clock cycle of the slowest of the two units. If the transmitter module runs faster than the receiver, then the transmitter module can output data in every clock cycle. Otherwise, it cannot output on an average more than one data item per receiver clock cycle. The *validT*data line is used by the transmitter module to indicate valid data items; invalid data items are ignored by the GRLS transmitter. If the transmitter runs faster than the receiver, the transmitter module may output bursts which are absorbed by the GRLS transmitter FIFO (see Fig. 7). The dimensioning of the buffer depends on the ratio between the frequencies and the characteristics of the transmitter module, i.e., the module generating the data items. The need for such buffer is common to all multifrequency interfaces and its dimensioning cannot be studied here because it depends on issues that are totally orthogonal to the topics of this paper. If it is known that the transmitter module outputs *valid* data items only seldom, the FIFO is not required. This is a fundamental difference compared to asynchronous FIFO GALS interfaces, where FIFOs are required for synchronization and not only for flow-control issues (see Section III). Relevant signals for a communication scenario with *NR* = 3 and *NT* = 2 are shown in Fig. 8. Data is output by the FIFOwhen *send* = 1 and held stable on the channel until the next



Algorithm 1 Regulation Algorithm

```
if $N_R \le N_T$ then
  $send = 1$
else
  $c_0 \Leftarrow N_R$
  $i \Leftarrow 0$
  loop
    Wait until the rising edge of $clk_T$.
    if $c_i > N_R - N_T$ then
      $send \Leftarrow 1$
      $c_{i+1} \Leftarrow c_i - (N_R - N_T)$
    else
      $send \Leftarrow 0$
      $c_{i+1} \Leftarrow c_i + N_T$
    end if
    $i \Leftarrow i + 1$
  end loop
end if
```

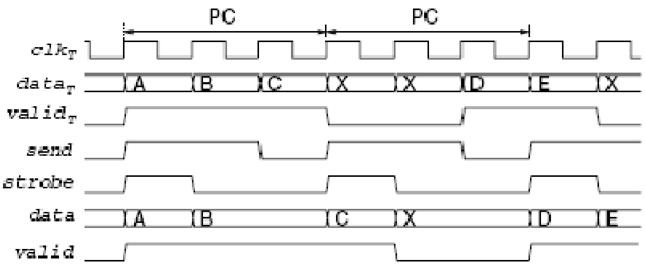


Fig. 8. GRLS transmitter signals.

clock edge in which *send* = 1. The interface utilizes zero wait state FIFOs, i.e., data items bypass the FIFO if the FIFO is empty and *send* = 1. If there is nothing to output when *send* = 1, then a dummy data item is output (shown as X in Fig. 8), with the additional *valid* line set to zero. At the receiver end of the channel, dummy data items are then discarded by the GRLS receiver. Whenever a data item is output, the *strobe* synchronization signal also toggles.The *send* signal is driven by the regulator (which is part ofthe transmitter as shown in Fig. 7), realized by Algorithm 1.The algorithm corresponds to the rate divider algorithmin The regulation algorithm is selected as it satisfiesa series of data-flow properties listed below, which arefundamental for the GRLS interface. Informally, a regulatorbased on algorithm 1 generates a periodic flow of data witha rate of min*(fT , fR)* introducing as little data-flow jitteras possible. More formally, a flow of data regulated byalgorithm 1 has the following properties.

1) *Average Rate:* The number of data items *d* output in a time *K TR* with *K* integer is always *d* ≤ *K* + 1.

2) *Periodicity:* The regulated flow of data is periodic with period PC: if a data item is output at time instant *τi*, another data item is also output at time instant *τi+PC= τi+ NRTT* .

3) *Minimal Instantaneous Rate:* The maximal amount of

time between two successive data outputs is *TM* ≤(*NR*/*NT*)*TT*

4) *Maximal Instantaneous Rate:* The minimal time between two successive data outputs is *Tm* ≥ *TR*/2 + *TH* /2.

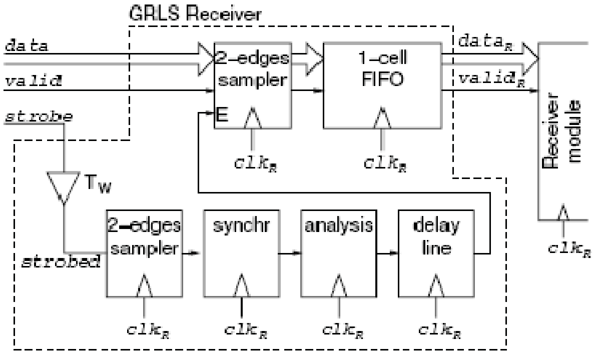


Fig. 9.GRLS receiver structure.

E. GRLS Receiver

The structure of the GRLS receiver is shown in Fig. 9.

The *strobe*signal is obtained by delaying the *strobe* signal by a delay *TW* using a delay line. *TW* is an arbitrary delay which corresponds to a fraction of *TH* (constraints on *TW* necessary for the interface to operate correctly are given later in this subsection). The *strobed*signal is sampled on every clock edge (positive and negative) of the receiver clock *clkR*. Some of the samplers may go metastable, i.e., there may be some setup or hold violations, and therefore the strobe samples are synchronized using high-latency multistage synchronizers before being analyzed (which gives time for the metastability to resolve itself), obtaining a sequence of samples denoted as

*s*0, *s*1, *s*2, . . . ,*si* , . . .

where*si*is the *strobed*value sampled at time *ti*(see Fig. 10). Because samples are obtained at half-cycle intervals, *ti*= *ti*−1 + *TR*/2. Denoting, respectively, as *t*su and *t*ho the setup and hold times of the strobe samplers, if the delayed strobe toggles between *ti*− *t*su and *ti*+ *t*ho, the strobe sampler may encounter metastability and, with a worst-case analysis, *si*stabilizes to a random value. In this case, we say that *si*is a corrupted sample. The minimal instantaneous rate dataflow property ensures that, as long as *TH*/2 >*t*su+ *t*ho, it is not possible to have two subsequent corrupted samples, i.e., if *si*is corrupted, then *si*−1 cannot be corrupted and vice-versa. Because of this property, if the GRLS receiver observes *si*_= *si*−1, it can conclude that the *strobed*signal transitioned between the time instants *ti*−1 − *t*su= *ti*− *TR*/2 − *t*su and *ti*+ *t*ho, as in Figs. 10(b) (*si*−1 is corrupted), 10(c) (*strobed* toggled between *ti*−1 + *t*ho and *ti*− *t*su), and 10(d) (*si*is corrupted). In fact, if the *strobed*signal did not toggle between *ti*−1 − *t*su and *ti*+ *t*ho, then none of the two strobe samples *si*−1 and *si*is corrupted and the two samples are necessarily equal [see Fig. 10(a)]. If *si*_= *si*−1 is observed by the GRLS receiver, then the GRLS receiver can conclude that the *strobed*signal transitioned between time instants *ti*−1 − *t*su and *ti*+ *t*ho. It can therefore conclude that the *strobe* signal transitioned betweentime instants *ti*−1 −*TW* −*t*su and *ti*−*TW* +*t*ho, and that a dataitem arrived at the same time. We remember here that *TW* denotes the delay between the *strobe* and the *strobed*signals. Based on the minimal instantaneous rate data-flow property, the data item was necessarily stable on the channel in the interval.

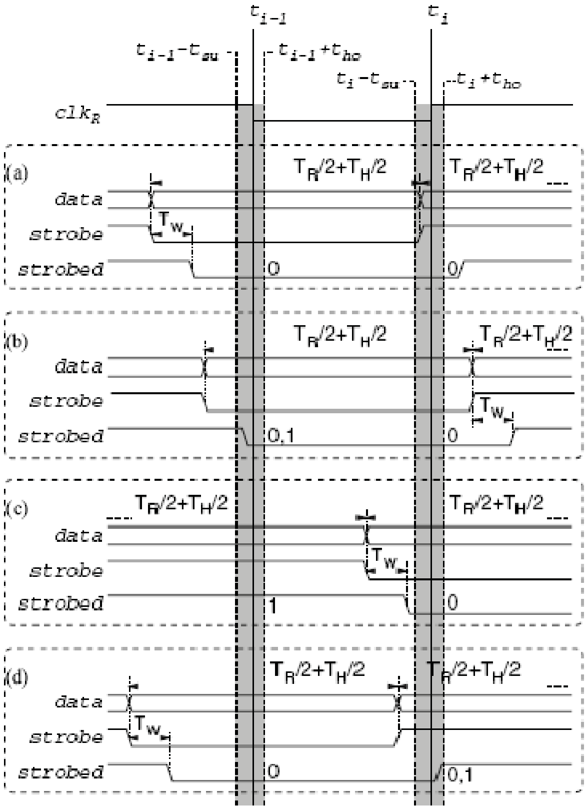


Fig. 10.Data/strobe transitions and strobe samples in a GRLS receiver. Possible values of the strobe samples are indicated near the sampling edges

$$\left(t_i - T_W + t_{\mathrm{ho}};\, t_{i-1} + \frac{T_R}{2} + \frac{T_H}{2} - T_W - t_{\mathrm{su}}\right) = \left(t_i - T_W + t_{\mathrm{ho}};\, t_i + \frac{T_H}{2} - T_W - t_{\mathrm{su}}\right).$$

As long as the following two properties hold:

$$T_W > t_{\mathrm{su}} + t_{\mathrm{ho}};\quad T_W < \frac{T_H}{2} - (t_{\mathrm{su}} + t_{\mathrm{ho}})$$

The data item could potentially have been safely sampled by the GRLS receiver at time instant *ti*, because it was stable during the metastability window of the sampler, i.e., between *ti*− *t*su and *ti*+ *t*ho.

Summarizing, if *si*_= *si*−1 is observed by the GRLS receiver, then the GRLS receiver concludes that a new data item could potentially have been safely sampled at time *ti*[Fig. 10(a) demonstrates that if *si*= *si*−1 data cannot necessarily be safely sampled on *ti*]. The analysis to reach this conclusion takes time, because it is necessary for the GRLS receiver to synchronize the *strobed*samples using high-latency multistage synchronizers beforeanalyzing them. When the analysis is complete, time instant *ti*has passed and the information would be useless if this was ageneric GALS communication scenario. However, the periodicity data-flow property combined with the periodic properties of rationally related systems ensure that the alignment between the receiver clock edges and data arrival time at the receiver end of the channel is periodic with period PC = *NT NRTH* (the least common multiple between the clock frequencies of the transmitter and the receiver), i.e., if a new data item could potentially have been safely sampled at time *ti*, a new data item can be safely sampled at time *ti*+ *J* PC = *ti*+2*J NT* , where *J* is an arbitrary integer value. In particular, it is guaranteed that a data item will be stable on the channel in the interval

$$\left(t_{i+2JN_T} - T_W + t_{\text{ho}}; t_{i+2JN_T} + \frac{T_H}{2} - T_W - t_{\text{su}}\right).$$

The GRLS receiver takes advantage of this, i.e., if it observes *si*_= *si*−1, it samples data at time *ti*+ *K* PC, where *K* is the smallest integer guaranteeing *ti*+ *K* PC ≥ *ta*, with *ta* being the time at which the analysis of the strobe samples obtained until time *ti*is completed. Since the strobe synchronizers are made up by a cascade of *NS* flip flops, then *ta* = *ti*+ *NSTR* and

$$K = \left\lceil \frac{N_S T_R}{PC} \right\rceil = \left\lceil \frac{N_S}{N_T} \right\rceil.$$

This solution ensures that the data item is sampled on the first available occasion after the *strobed*samples analysis is completed. Thus, the strobe analysis stage performs a continuous, adaptive learning phase. Although double-stage synchronizers are normally very safe to raise mean time between failures (MTBF), the number *NS* of stages in the multistage synchronizers can be raised. Unlike asynchronous FIFOs, the number of synchronization stages affects the length of the learning phase but does not affect data latency. The GRLS interface operation is shown in Fig. 11(a) and (b) (the *valid* signal is omitted to keep the figure simple).Fig. 11(a) shows a scenario where no metastability arises while sampling the *strobed*signal. Data is sampled *K* PC = 2 PC = 6*TR* after all the time instants in which was obtained a *strobed*sample that was different compared with the sample obtained half a cycle earlier. As long as *TH/*2 >*t*su+ *t*ho, the minimal instantaneous rate data-flow property ensures that every strobe transition is detected (every time the *strobed*signal toggles, the new value remains stable on the channel sufficiently long to be sampled in a metastability-free fashion at least once). Thus, the mechanism guarantees that all data items are sampled and that no data item is sampled twice. Fig. 11(b) shows a similar scenario, but the *strobed*sample obtained on clock edge *R* is corrupted because a *strobed*transition close to that clock edge violates the metastability window of the *strobed*sampler. Depending on how the *strobed*sample stabilizes, data sampling can happen on clock edge *A*or on clock edge *B*.Which one is selected is irrelevant because both are safe for data sampling. The alignment between *data*, *strobe*, *strobed*, and *clk R* is identical around clock edges *R*and *A*. Since *strobed*violated the metastability window of the *strobed*sampler on clock edge *R*, it violates also the metastability window of the *strobed*sampler on clock edge *A*. This means that the *data* lines, whose transitions happen a time*TW*before *strobed*transitions, cannot violate the metastabilitywindow of the data sampler on clock edge A as long as *TW* >*t*su+ *t*ho. Similarly, since it is guaranteed that two data transitions cannot happen closer than *TR/*2 + *TH/*2, the data lines are guaranteed to remain stable until after the end of the metastability window of the data sampler on clock edge *B* as long as *TW* < *TH /*2 − *(t*su+ *t*ho*)*, and clock edge *B* is also safe for data sampling. As Fig. 11(a) shows, it is possible that the GRLS receiver needs to sample two data items in a single clock cycle, one on the positive edge of the clock and one on the negative edge of the clock, such as on clock edges A and B. In the same time, the receiver module can only consume a single data item. Buffering is therefore needed to store the additional data item in the GRLS receiver until it can be consumed. However, the average rate data-flow property ensures that in *C* clock cycles of the receiver at most *C* + 1 data items can be received. In the same interval, the receiver module consumes *C* data items. Therefore, a single-cell FIFO buffer is sufficient to hold the remaining data item, and will never overflow.

III. RELATED WORK

This section reviews the main techniques that are usedto build GALS and mesochronous interfaces, as well as the techniques that were previously introduced to synchronize data between two rationally related clock domains, to highlight similarities and differences compared with the GRLS communication scheme that is illustrated.

A.GALS Interfaces

Pausible-clock techniques require the use of stoppable ring oscillators driven by a MUTEX-based asynchronous handshake mechanism. Pausible-clock techniques guarantee infinite MTBF but the clocks might be stopped for a potentially unbounded amount of time. The technique requires components that are not normally found in standard technology libraries, which has prevented it from becoming an established industrial practice, remaining mostly confined to research and niche applications Clock-gating techniques are an evolution of passible-clock interfaces that employ a standard-cells-only clock-gating mechanism implemented as an asynchronous state machine to stop an external clock source when communication takes place. Clocks cannot be stopped for more than an arbitrary number of cycles; MTBF is not infinite but is typically very high Both Passible-clock and clock-gating interfaces require that the clock of a whole module is stopped when communication takes place on any of its ports, which can introduce heavy performance penalties.

For both interfaces, no more than one data item can be transmitted per two handshake round-trips, which limits both latency and throughput. For example, if a channel has a 500ps propagation delay, even with perfect control no more than one data item can be transmitted every 2 ns. Reference analyzed and compared several different synchronization methods for GALS systems and proposed local delay latching, a technique that inserted latches on input/output ports to allow communication to safely take place between two unrelated clock domains. The technique requireMUTEXes to generate the latch control signals and is shown to work well when the clock period is relatively slow compared with the propagation delays of the technology. It also employs handshake, i.e., the performances are also limited by the round-trip delay. Asynchronous FIFOs have become the standard solution for clock-domain crossing in GALS systems as they are easily compatible with today's application-specific integrated circuit design flow. Because of their success, they are taken as a reference in this paper for comparison with GRLS. Unlike passible-clock and clock-gating interfaces, they can be designed at RTL and synthesized. They are internally synchronized ,i.e., they contain a cascade of registers to synchronize the Grey write pointer to the receiver clock domain. The internal synchronization mechanism determines the latency of the interface.

B. Mesochronous Interfaces

The most widely used mesochronous interfaces are based on the STARI approach and use self-timed FIFOs which are initialized to be half-full. In one clock cycle, the transmitter writes one data item and the receiver reads another item, avoiding overflows and underflows. Four-elements FIFOs are normally used. When initialized properly, overflow and underflow never occur because the clock frequencies are matched. STARI solutions can tolerate a high jitter between the clocks but introduce a two-cycle latency. Learning-phase mesochronous solutions such as STSS and SKIL target low-latency communication scenarios. A clock edge (positive or negative) is selected during a learning phase, and data is always sampled on that edge (at least one of the two is always guaranteed to be safe for data sampling because the clock frequencies are perfectly matched). The learning phase can happen only once upon reset or continuously during operation. Learning-phase interfaces guarantee low latency and are the most direct source of inspiration for the GRLS interface.

C.Ratiochronous Interface

There have been previous attempts at designing communication interfaces specifically tailored for rationally related frequencies by building on the properties of rationally related systems analyzed in Sections II and III.

The earliest attempt is the rational clocking interface in which assumes no phase difference between the clocks and uses only the positive edge of the receiver clock for data sampling. To guarantee maximal throughput, the receiver alternates between two separate registers sets, which leads to a “lazy” algorithm and no optimal latency. The rational clocking interface cannot solve the GRLS communication problem because it requires a known skew among the clocks. In a protocol-aware formalism is introduced to calculate in which cycles synchronization failures can arise, improving latency and overhead figures for the rational clocking interface. However, the interface in suffers from the same limitations of the rational clocking approach. Also, when the phase difference between the transmitter and the receiver clocks is unknown, a worst case analysis is the only feasible approach and this research cannot be applied to solve theGRLS communication problem.

One interface that targets specifically the GRLS communication problem isIn this paper, the STARI approach is generalized by reducing the size of the FIFO to a single handshaking stage realized with a latch. Three latches areintroduced on the data path: the first is controlled by the transmitter clock, the last by the receiver clock and the central becomes transparent only at specific time instants in which data can safely cross the clock domain boundary. Controlling the central latch is the main challenge of this approach, and the solution proposed by the authors relies on complex transistor-level design, which makes it an unlikely solution for commercial applications. This solution is based on totally different concepts compared to ours but obtains the same latency figures, albeit with higher design-flow complexity because of the nonstandard components.

The work in is a fully digital solution that estimates the phase difference between data arrival time and receiver clock edges using a phase estimator block, in which the phase difference is expressed as a fractional number and is continuously updated based on the knowledge about the clock frequencies and analysis of the incoming flow of data items. As for GRLS, a learning phase is used, i.e., the phase difference is estimated in advance and used later to determine if data should be sampled using the rising or the falling edge of the receiver clock. However, the phase estimation block is more complex compared with the one of the GRLS interface as it requires the ability to perform arithmetic operations.

IV.IMPLEMENTATION

The detailed standard-cells implementation of a completeGRLS interface is shown in Fig. 12. Save for the delay line, the GRLS interface is a synthesizable RTL design. The delay line can also be realized using a cascade of standard-cell buffers. The implementation of the GRLS transmitter is straightforward. Data, *valid* and strobe lines are routed together.

In the GRLS receiver, a delay line is inserted on the strobe path and two flip-flops, one positive- and one negative-edge-triggered, are used to sample the delayed strobe continuously. The strobe samples are synchronized using cascades of flip-flops and then compared with the sample arrived half a cycle earlier using a couple of XOR gates. A programmable cascaded delay line made by a cascade of flip-flops follows. The selector value is determined as $KNT - NS - 1$ where NS is the number of synchronization stages and K is chosen as the smallest integer guaranteeing $KNT - NS - 1 \geq 0$. The delay line ensures that the time interval between sampling of the strobed signal and data sampling is the smallest possible multiple of PC. The cascaded delay line must contain at least $KNT\text{max} - NS - 1$, with $NT\text{max}$ being the maximal value NT can take. The *sp*and *sn*signals generated in the strobe analysis stage of the GRLS receiver drive data sampling, i.e., data is sampled only when the strobe analysis stage has determined that it is useful and safe to do so.

This solution ensures that metastability never enters the data path and remains confined to the strobe analysis stage of the GRLS receiver, where high latency synchronizers ensure a high MTBF.

No synchronizers are inserted on the data path, which ensures that the number of strobe synchronization stages *NS* is independent compared with data latency. If *sp*= 0 (*sn*= 0) on a positive (negative) clock edge, then *v p* (*vnz*) is cleared, otherwise the value of the*valid* signal is stored in *v p* (*vnz*) and the value of the data signal is stored in *dp*(*dnz*). The *dnz*and *vnz*signals are synchronized to the receiver clock domain. *v p* (*vn*) indicates that a *valid* data item is just sampled on the positive (negative) edge of the clock. The *ds* register acts as a one-cell FIFO to absorb the bursts ofdata sampled on two consecutive edges. When two data items are contained in the registers *dp*and *dn* (*v p* = *vn*= 1), the oldest (*dn*) is output and the newest is saved in the *ds* register. *vs*is set to one when the *ds* register contains a *valid* data item. When one *valid* data item is contained in the *ds* register (*vs*= 1) and one *valid* data item is contained

V. **A**REA OVERHEAD AND COMPLEXITY

The GRLS interface is composed of standard cells only.Except for one delay line, the rest of the interface can be designed in a high-level language (RTL) and synthesized for any technology. Thus, the standard design flow need of the VLSI industry is satisfied. Area overhead analysis follows. Excluding the transmitter FIFO (shown in Fig. 12) from the analysis (it is required in all multifrequency interfaces and its size depends on flow-control considerations that fall outside the scope of this paper), the GRLS transmitter and receiver require a number of flip-flops equal to_log2 *(N*max*)* + 4 *(W* + 1*)* + 4 + 2*NS* + 2 *(N*max− *NS* − 1*)*where *W* is the number of data lines and *N*max the maximaldivision ratio. As the number of data lines grows, the area overhead of the interface tends to four flip-flops per data line. The area overhead of asynchronous FIFOs as implemented in is hard to analyze because the size of the FIFO depends on both synchronization and flow-control issues. The FIFO size must be a power of two; eight cells FIFOs are necessary to guarantee maximal throughput in worst-case scenarios MesochronousSTARI and STSS interfaces also require 4 flip-flops per data line.

VI. **R**OBUSTNESS

In Section II, two constraints necessary for the GRLSinterface to work were introduced

$$T_W > t_{\text{su}} + t_{\text{ho}}; \quad T_W < \frac{T_H}{2} - (t_{\text{su}} + t_{\text{ho}}) .$$

The constraints are meant for an ideal scenario; in reality, in addition to the setup and hold time, the synchronization interface will have to cope with jitters, propagation delay misalignments between data and strobe, and the potential variation over time of these no idealities. The GRLS interface is capable of coping with these nonidealities if *fH*is under a certain bound, which can be calculated based on the worst-caseno idealities. We consider the following nonidealities parameters.

1) The arrival time of each data and strobe line to the GRL receiver is subject to a maximal jitter *JT* compared tothe ideal case. The jitter takes into account the jitterof the transmitter clock and the jitter added during thepropagation through the channel.

2) The time in which the receiver clock edges occur is subject to a maximal jitter *JR*.

3) The maximal misalignment between the propagationdelay of a data line and the strobe line is MIS.

The GRLS interface builds on the assumption that, when a strobe transition is detected at time *ti*, then the data lines are stable between (1)

$$\left(t_{i+2KN_T} - T_W + t_{\text{ho}}; t_{i+2KN_T} + \frac{T_H}{2} - T_W - t_{\text{su}}\right).$$

In presence of nonidealities, the window over which it is guaranteed that the item is stable is reduced to

$$\begin{aligned}(t_{i+2KN_T} &- T_W + J_R + J_T + \text{MIS} + t_{\text{ho}}; \\ &t_{i+2KN_T} + \frac{T_H}{2} - T_W - J_R - J_T - \text{MIS} - t_{\text{su}}).\end{aligned}$$

To guarantee that the interface operates correctly, the interval must not overlap with the metastability window of the data sampler at *ti*+2*KNT* , i.e., with the interval *ti*+2*KNT* −*t*su;+*ti*+2*KNT* + *t*ho_. To guarantee that this happens, the following two relations must hold:

$$\begin{aligned}T_{\text{Wmin}} &= J_R + J_T + \text{MIS} + t_{\text{su}} + t_{\text{ho}} \\ T_{\text{Wmax}} &= \frac{T_H}{2} - J_R - J_T - \text{MIS} - t_{\text{su}} - t_{\text{ho}}.\end{aligned}$$

The two relations, coupled with the impossibility to build perfect delay lines, determine the maximum value for the least common multiple *fH*= 1/*TH* between the frequencies involved in communication. Even if it was possible to create perfect delay lines, i.e., if the delay line was built with *T*Wmin= *T*Wmax= *JR* + *JT* + MIS + *t*su+ *t*ho, the relations would give the following bound for *fH*:

$$f_{\text{Hmax}} = \frac{1}{4\left(J_R + J_T + \text{MIS} + t_{\text{su}} + t_{\text{ho}}\right)}.$$

As an example, let us consider a 90-nm implementation of a basic GRLS system. The data jitter *JT* is given by *JT* = *JH* + *JC* + *JP* = 60 ps, where *JH* = 20 ps is the jitter of the global clock, *JC* = 20 ps is the jitter introduced by the LCGU of the transmitter, and *JP* = 20 ps is the jitter of the propagation delay through the channel caused by crosstalk. The receiver clock jitter *JR* is given by *JR* = *JH*+*JC* = 40 ps. Let us consider also a misalignment between the data and strobe lines MIS = 50 ps, and let us consider *t*su+*t*ho= 40 ps. With *fH*= 1 GHz, the value of *TW* is constrained to be between 190 and 310 ps, which is easily manufacturable. As long as all parameters are within the given range, it is formally proven that the interface operates correctly.

We point out that it is in practice hard to determine values for MIS and that the jitter values that should be used are the maximal jitters between clock cycles that fall at an interval *K*PC from each other (the jitter introduced between the time in which the strobe transition takes place and the time in which the data is sampled).With small values of *NT* and *NR*, PC is in the order of around 1–10 receiver/transmitter cycles and jitter values might be consistent with short-term jitter values, but with higher values of *NT* and *NR* long-term jitter values should be considered. This introduces a limitation to the maximal value of *NT* and *NR*.

VII. LATENCY ANALYSIS

In a globally nonsynchronous interface, as opposed to a synchronous interface, the performances are not deterministic and depend on the skew among the clocks. Therefore, best case, average-case, and worst case latencies can all be defined. Latency can be measured in terms of receiver clock cycles, and is measured in absence of contention, i.e., when all buffers are empty and a single data item travels from the transmitter to the receiver. We compare the latency of a GRLS interface with that of asynchronous FIFO GALS and learning-phase mesochronous interfaces. For all three interfaces, we assume
*t*su_ *t*ho_ 0 and a null channel propagation delay to make the discussion more general. This does not make the analysis biased because the channel propagation delay, as well as the setup and hold times, have the same impact on all three interfaces.The best-case, average-case, and worst case latencies ofasynchronous FIFO GALS interfaces are given by

*L*AF,BC= 2*TR*; *L*AF,AC = 2.5*TR*; *L*AF,WC = 3*TR*.

The fastest learning-phase mesochronous interfaces, such as, introduce a latency of

$$\begin{aligned} L_{\mathrm{LPM,BC}} &= 0 + T_K \\ L_{\mathrm{LPM,AC}} &= 0.5T_R + T_K \\ L_{\mathrm{LPM,WC}} &= T_R + T_K. \end{aligned}$$

The parameter *TK* is a fraction of the clock cycle, whichdepends on the type of interface. It can be set as a tradeoff between tolerance to nonidealities and latency. Tolerance to nonidealities of a mesochronous interface with a given *TK*is equal to that of a GRLS interface operating in the same conditions with *TW* = *TK* . The latency of a GRLS interface can be broken down into different components

*L*GRLS = *LT* + *LS* + *LR*

where *LT* is the transmitter latency (determined by the regulation algorithm), i.e., the time it takes for one data item to cross the GRLS transmitter, *LS* is the latency introduced by the skew between the clocks, and *LR* is the latency introduced by the GRLS receiver. *LS* and *LR* are as follows

*L*S,BC= 0; *L*S,AC = 0.5*TR*; *L*S,WC = *TR*
*L*R,BC= *LR,AC* = *L*R,WC = *TW*.:

The best-case transmitter latency is *L*T,BC= 0 because a data item that is output by the transmitter module when *send* = 1 is output immediately. Based on the minimal instantaneous rate data-flow property, which establishes the maximum time between two data outputs, the worst case transmitter latency is given by

$$L_{\mathrm{T,WC}} = \left(\left\lceil \frac{N_R}{N_T} \right\rceil - 1\right) T_T$$

as this is the maximum time a data item can be blocked in the GRLS transmitter if the FIFO is empty. *L*T,AC can be determined by averaging the hypothetical transmitter latencies of data items output by the transmitter module in every single cycle of a periodicity cycle. By considering the three components, the latency of a GRLS interface can be calculated as

$$\begin{aligned} L_{\mathrm{GRLS,BC}} &= T_W \\ L_{\mathrm{GRLS,AC}} &= L_{\mathrm{T,AC}} + \frac{T_R}{2} + T_W \\ L_{\mathrm{GRLS,WC}} &= \left(\left\lceil \frac{N_R}{N_T} \right\rceil - 1\right) T_T + T_R + T_W. \end{aligned}$$

The following bounds can be extracted for the average-case and worst case latency of a GRLS interface with *TW* = 0

$$\frac{T_R}{2} \le L_{\mathrm{GRLS,AC}} \le T_R; \qquad T_R \le L_{\mathrm{GRLS,WC}} \le 2T_R.$$

TW is chosen as a trade-off between latency and robustness, with the most robust value being *TW* = *TH/*4 and the value giving best latency but no tolerance to nonidealities being *TW* = 0. As the impact of *TW* on the latency is small (only 0.25*TR* in the worst case, when *TR* = *TH*), it is recommended to dimension the delay line with *TW* = *TH* /4.
Learning-phase mesochronous interfaces require having *fR*= *fT*⇒*NR* = *NT* . For those situations, the GRLS transmitter outputs data in every clock cycle and the transmitter latency is null: the latency of the GRLS interface is identical to that of the equivalent learning-phase mesochronous interface with *TK* = *TW*, which has also the same tolerance to nonidealities. Despite the increased flexibility, the GRLS and mesochronous interfaces have identical performances. Based on the latency bounds of the GRLS interface, the following bounds can be calculated for a GRLS interface with *TW* = 0.

$$\begin{aligned} \frac{1}{5} L_{\mathrm{AF,AC}} &\le L_{\mathrm{GRLS,AC}} \le \frac{2}{5} L_{\mathrm{AF,AC}} \\ \frac{1}{3} L_{\mathrm{AF,WC}} &\le L_{\mathrm{GRLS,WC}} \le \frac{2}{3} L_{\mathrm{AF,WC}}. \end{aligned}$$

In Table I, the latencies of GRLS and GALS interfacesare compared together given all possible combinations of *NT* and *NR* between 1 and 8. For GRLS, *TW* = *TH/*4 is assumed (the most robust choice). Comparing the average-case and worst case latencies of GRLS across the Table I with the average-case and the worst case latencies of asynchronous FIFO GALS interfaces (respectively 2.5*TR* and 3*TR*), the highest latency improvements are, respectively, 78.8% and 65.6%, whereas the lowest latency improvements are, respectively, 60.8% and 36.4%. The average latency improvements over the whole table are, respectively, 71.8% and 53.3%. By switching from the GALS design style to the GRLS design style, communication performances can thus be improved in average _72%, i.e., communication latency is nearly cut by a factor 4.

VIII. CONCLUSION

This paper introduced a low-latency communication interface for multifrequency links, which introduced a muchsmaller performance overhead compared with state-of-heartGALS interfaces. The interface can be designed at RTL except for a single delay line and hds an overhead of four flip-flops per data line as for the state-of-the-art mesochronous interfaces. Its performances were close to those of the fastest mesochronous interfaces but it supported nodes running at different frequencies (with a ratiochronous constraint). The GRLS interface had a good tolerance against nonidealities and automatically adapted to changes in the skew between the clocks. It gave a 4 average latency improvement over the state-of-the-art asynchronous FIFO GALS interfaces. Unfortunately, our worst case nonidealities analysis showed that the interface can work well in 90-nm technology if the least common multiple between the transmitter and the receiver clock frequencies is at most 1 GHz, which limits its applications to systems running at relatively low frequencies and/or in which the ratio between the transmitter and the receiver clock frequencies has both a small numerator and a small denominator. For those systems, however, the GRLS interface makes the GRLS design style well suited to meet current needs of VLSI industry.

REFERENCES

[1]. J. M. Chabloz, “Distributed DVFS with rationally-related frequencies and quantized voltage levels,” in *Proc. Int. Symp. Low-Power Electron.Design*, Aug. 2010, pp. 247–252.

[2]. M. N. Horak, S. M. Nowick, M. Carlberg, and U. Vishkin, “A lowoverhead asynchronous interconnection network for GALS chip multiprocessors,” in *Proc. Int. Symp. Netw. Chip*, May 2010, pp. 43–50.

[3]. J. M. Chabloz and A. Hemani, “A flexible interface for rationally-related frequencies,” in *Proc. Int. Conf. Circuit Design*, Oct. 2009, pp. 109–116.

[4]. J. M. Chabloz and A. Hemani, “Lowering the latency of interfaces for rationally-related frequencies,” in *Proc. Int. Conf. Circuit Design*, Oct. 2010, pp. 23–30.

[5]. W. J. Dally and S. G. Tell, “The even/odd synchronizer: A fast, alldigital, periodic synchronizer,” in *Proc. Int. Symp. Asynchron. Circuits Syst.*, May 2010, pp. 75–84.