A Flexible Communication Scheme for Rationally-Related Clock Frequencies

Jean-Michel Chabloz and Ahmed Hemani
Department of Electronic Systems, School of ICT, KTH - Royal Institute of Technology, Stockholm
Email:{chabloz,hemani}@kth.se

***Abstract*—As a replacement for the fast-fading Globally-Synchronous model, we have defined a flexible design style for SoCs, called GRLS, for Globally-Ratiochronous, Locally-Synchronous, which does not rely on global synchronization and is based on using rationally-related clock frequencies derived from the same source. In this paper, using the special periodical properties of rationally-related systems, we build a latency-insensitive, maximal-throughput, low-overhead communication method, based on the idea of using both clock edges to sample data at the Receiver. The validity of the method and its resistance to non-idealities such as jitter, misalignments and clock drifts are formally proven while experimental results including overhead are presented for 90 nm technology. Despite allowing much greater flexibility, the overhead of our method is comparable to that of state-of-the-art mesochronous communication techniques. We also show performances, complexity and overhead improvements over all other approaches that have so far been proposed for rationally-related clock frequencies.**

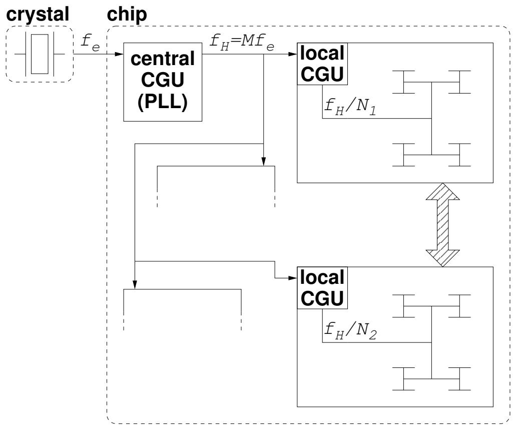

Fig. 1. Ideal layout of a GRLS system

I. Introduction

With technology scaling, designing globally balanced clock trees is a difficult proposition [1] and any small change in design requires timing closure iterations that aggravate the SoC engineering costs. What is needed is a latency-insensitive design style that will enable hierarchical physical design with little or no chip-level timing closure issues. Multiple clock domains are routine in today's SoCs [2]. Eliminating the global clock tree leads to easier timing closure, higher working frequencies and a considerable reduction in the power consumption of the clock distribution net [3]. Several taxonomies of non-fully-synchronous design styles have been proposed [2], [4]. GMLS, for *Globally-Mesochronous, Locally-Synchronous*, uses a single clock frequency that is distributed throughout the chip using a global unbalanced distribution tree coupled with local balanced clock trees [5]. Recently there has been a lot of interest around this design style [6]–[8]. More generally, GALS systems, for *Globally-Asynchronous, Locally-Synchronous* [9]–[11], a term that we use here with a broad meaning, consists in using totally unrelated clock frequencies in the different synchronous islands.

We are investigating the concept of building chips based exclusively on rationally-related frequencies obtained from the same source and non skew-aligned. We call this design style GRLS, for *Globally-Ratiochronous, Locally-Synchronous*.

A conceptual view of the clock distribution scheme in a GRLS chip is shown in Figure 1. A single high-frequency clock generated by a central Clock Generation Unit (CGU) is distributed throughout the GRLS chip. This global clock net has a low fanout, being equal to the number of synchronous islands, and is unbalanced. No assumptions are made on the phase relationship of the clock edges at the leaf nodes of the global clock tree. A local CGU in every synchronous island divides the frequency of the high-frequency clock by a programmable factor. Thus, in a GRLS system all local clocks have rationally-related frequencies and are not skew-aligned.

The flexibility of GRLS in the choice of the local frequencies lies between those of GMLS and GALS, as the frequencies on the chip are constrained to be submultiples of the same global frequency. However, local CGUs in a GRLS system are very simple, fast and easily-programmable fully-digital blocks. This adds flexibility to the system, and allows a very easy and fast implementation of Distributed Frequency Scaling [12], which in turns could help decrease the total power consumption of the system. In this respect, the loss of flexibility compared to GALS is mostly theoretical, as programmable local oscillators in a GALS system are usually obtained by using clock dividers and are therefore only discretely programmable [13]. Because all clocks are derived from the same source, GRLS also guarantees the same frequency stability of GMLS systems. Integrating heterogeneous IPs with different frequency requirements is also more simple in a GRLS system compared to a GMLS system.

Compared to GALS, an additional overhead for the distribution and division of the global clock is introduced. A simple example of a very fine-grained GRLS chip with a global 1 GHz clock was carried out for a $4\,mm^2$ chip in 90 nm technology with 25 synchronous islands, each hosting a local CGU (an unoptimized programmable divider for ratios between 1 and 64) driving 3600 flipflops, roughly corresponding to a simple RISC processor. Two scenarios were analyzed, one with an average island frequency of 500 MHz and the other with an average frequency of 125 MHz. The power breakdowns for the two cases are shown in figure 2. As expected, the percentage overhead is higher when the average island frequency is lower, which translates to lower power consumption in the local clock net. The analysis shows that the additional power overhead of GRLS compared to GALS and GMLS is small even for extremely fine-grained systems, unlikely to happen in reality, and can be outweighted by the benefits of distributed Dynamic Voltage Frequency Scaling.

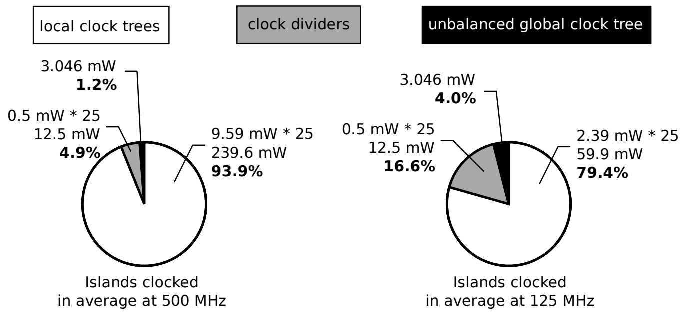

Fig. 2. Power breakdown of a $2\,mm \times 2\,mm$ GRLS system with 25 equal-sized islands, each containing 3600 flipflops, using a 1 GHz global clock.

The contribution of this paper consists in defining, building on the special periodic properties of rationally-related clock frequencies, a maximal-throughput, low-latency, low-overhead communication method for GRLS systems which can be inserted at high-level and dynamically adapts to any phase difference between the clocks. Despite the increased flexibility in the choice of the local frequencies, performances and overhead are comparable to state-of-the-art mesochronous interfaces. We also show performances, overhead, flexibility and complexity improvements over all existing techniques that have so far been proposed to cross the boundary between two rationally-related clock domains.

The remainder of the paper is organized as follows: in section II, a survey of existing approaches is presented. Section III formally defines the communication problem in a GRLS system and introduces the notation used in the rest of the paper. Section IV presents our communication scheme in detail. Section V analyzes how the scheme can cope with non-idealities such as jitter, strobe and data misalignments and clock drifts. In section VI we compare our interface with other approaches in terms of throughput, latency and overhead. In section VII we gauge the overhead of our scheme in 90 nm technology and calculate a realistic upper bound for the working frequency of the system. Section VIII draws conclusion and analyzes future work.

II. Related Work

GALS communication methods such as pausible-clocking [10] and asynchronous FIFOs [11] can be used to cross the boundary between rationally-related clock frequencies. Compared to pausible-clocking, our interface avoids the handshake latency, the performance penalty for communication and the inherent non-determinism; compared to asynchronous FIFOs, we avoid synchronization latency.

There have been previous attempts at designing interfaces for rationally-related frequencies. The earliest research was carried out by Sarmenta and coworkers at MIT [14]. For their *Rational Clocking* interfaces, the authors assume no phase difference between the clocks. As the authors use only the positive edge of the clock for sampling data at the Receiver, their fast-Transmitter interface can guarantee maximum throughput only by using two sets of output registers in the Transmitter. In different clock cycles, the Receiver should alternate between the two registers, which leads to a lazy algorithm with non-optimal latency.

We argue that in modern-day systems it is difficult to keep the clocks aligned, as the clock trees are large and their delay can easily exceed the clock period. For example, in the same $400\ \mu m \times 400\ \mu m$ synchronous island containing 3600 synchronous elements that was considered in section I, the delay of the local clock tree can be as high as $1.5\ ns$, which is comparable with the minimal clock period supported by a small RISC core. Even if the designer managed to have phase-aligned clocks at the root of the local clock trees, the propagation delay through the local trees would break the alignment. To get aligned clocks, the global clock tree should compensate for the delays of the local clock trees, which would increase the power consumption in the global clock tree (55% in our example), prohibit hierarchical physical design and complicate the design flow.

Arguing that Sarmenta's approach is essentially a worst-case analysis, in [15] a protocol-aware formalism is introduced to calculate in which cycles synchronization failures could arise, ameliorating latency and overhead figures of the communication scheme. However, when the phase difference between the Transmitter and the Receiver clocks is unknown, a worst-case analysis such as the one presented by Sarmenta is the only feasible approach.

We also compare our approach to what is perhaps the most obvious solution to the problem of interfacing two clock domains with rationally-related clock frequencies and unknown skew. A lot of research effort has been lately put into developing synchronizers for mesochronous communication. If a high-frequency unit clocked with a multiple of the two frequencies is inserted in the channel, two rationally-related clock domains can be interfaced using a double mesochronous link. State-of-the-art solutions for mesochronous communication are based on the STARI approach [5] and use self-timed FIFOs which are initialized to be half-full. In one clock cycle, the transmitter writes one item of data and the Receiver reads another item, avoiding overflows and underflows. Four-elements FIFOs, which can be realized using latches, result in a small overhead and have become some sort of standard solution in industrial and state-of-the-art applications [6], [7]. Other mesochronous solutions, such as STSS [16] and SKIL [17], are more similar to our approach as they propose to dynamically select the edge of the clock used at the Receiver for sampling data.

In [18], the STARI approach was generalized by reducing the size of the FIFO to a single handshaking stage realized with a latch. This leads to a optimal-latency communication scheme introducing three latches on the data path: the first controlled by the Transmitter clock, the last controlled by the Receiver clock and the central becoming 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. The authors also propose an interface for rationally-related clock frequencies, using a rate multiplier to divide the rate of the faster clock. This solution obtains the same latency figures of our approach, but its high design complexity makes it an unlikely solution for commercial applications.

III. Problem Definition and Notation

We propose a communication scheme to interface a Transmitter clocked by a clock clk_T at frequency $f_T = \frac{1}{T_T}$ and a Receiver clocked by a clock clk_R at frequency $f_R = \frac{1}{T_R}$ under the constraint that there exists an ideal or physical clock with frequency $f_H = lcm\,(f_T, f_R) = \frac{1}{T_H}$. When $f_R > f_T$, we call the system *fast-Receiver*; when $f_T > f_R$, we call the system *fast-Transmitter*; If $f_R = f_T$, the system is *mesochronous*. We make the assumption to have a stable, albeit unknown, phase difference between the clocks. We will show in section V how this constraint can be largely relaxed.

We define $N_T = \frac{f_H}{f_T}$ and $N_R = \frac{f_H}{f_R}$.

We further define $P = lcm\,(N_T, N_R)$. The periodicity cycle $PC = P\,T_H$ is the smallest common time interval that both Transmitter and Receiver clocks can count, $\frac{P}{N_T}\ clk_T$ ticks and $\frac{P}{N_R}\ clk_R$ ticks respectively.

It can be noted that any communication method for rationally-related clock frequencies poses, explicitly or implicitly, some limitations on f_H. Clearly, when the least common multiple of f_T and f_R is higher than a certain bound, for all practical purposes they should be considered unrelated.

IV. The GRLS Communication Interface

In this section we describe in detail the GRLS communication scheme. We begin by selecting a flow control algorithm, whose properties, analyzed in subsection IV-A, are the basis for the GRLS communication scheme. In subsection IV-B we present the GRLS Receiver Interface. Finally, in subsection IV-C, we analyze simplifications of our communication scheme for fast-Receiver/mesochronous systems and situations in which the phase difference between the clocks is known.

A. Flow Control

Two systems cannot communicate, on average, faster than the slower of the two clock frequencies. In fast-Transmitter systems, bursts need to be absorbed by FIFOs, whose dimensioning is not within the scope of this paper. A flow control algorithm decides on which transmitter clock edges data is transmitted (see figure 3). We select the rate divider algorithm in [18] as our control-flow algorithm, as its properties are crucial to the GRLS communication scheme. The pseudo-code is shown as Algorithm 1. Figure 3 shows the application of the regulation algorithm to a fast-Transmitter scheme. The GRLS communication scheme is a form of source-synchronous communication using a $strobe$ signal that toggles whenever $send$ is asserted. A new data item can be output only when the strobe toggles. If the data item is a valid item, the $valid_c$ signal is set to one; if the Receiver has nothing to send, the $valid_c$ signal is set to zero.

Algorithm 1 can handle all three clocking scenarios. However, if the system is mesochronous or fast-Receiver, data is output in every clock cycle and the regulator is redundant.

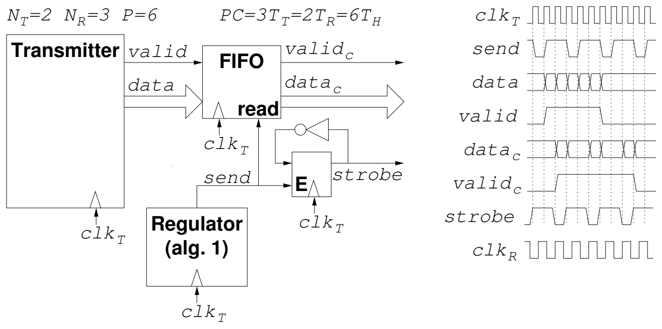

Fig. 3. Data regulation at the Transmitter

Algorithm 1 Regulation algorithm

```
1:  if $N_R \leq N_T$ then
2:      $send = 1$
3:  else
4:      $c \Leftarrow N_R$;
5:      loop
6:          Wait until the rising edge of $clk_T$.
7:          if $c > N_R - N_T$ then
8:              $send = 1$
9:              $c \Leftarrow c - (N_R - N_T)$
10:         else
11:             $send = 0$
12:             $c \Leftarrow c + N_T$
13:         end if
14:     end loop
15: end if
```

Informally, a regulator based on algorithm 1 generates a periodic flow of data with a rate of $min\,(f_T, f_R)$ introducing as little jitter as possible. More formally, a flow of data regulated by algorithm 1 has the following properties, whose formal proofs are being skipped to meet the space constraints:

1) *average rate:* The number of data items d output in a time $K\,T_R$ with K integer is always $d \leq K+1$, with $K \leq d \leq K+1$ if the Transmitter is sending data at the maximum rate in a fast-Transmitter system.
2) *periodicity:* The regulated flow of data is periodic with period PC: if data can be output at time instant t_i, data can also be output at time instant $t_i + PC$.
3) *maximal instantaneous data rate:* The minimal time between two successive data outputs is $T_D > \frac{T_R}{2}$. Since data is output only on positive clk_T edges, T_D is a multiple of T_T, which is a multiple of T_H. Since $\frac{T_R}{2}$ is a multiple of $\frac{T_H}{2}$, $T_D \geq \frac{T_R}{2} + \frac{T_H}{2}$.
4) *minimal instantaneous data rate:* Assuming that the Transmitter is sending data at the maximum possible rate, the maximal amount of time between two successive data outputs is $T_T \left\lceil \frac{N_R}{N_T} \right\rceil$.

Algorithm 1 can be implemented as an FSM, but for small values of N_T and N_R, a more optimal solution is to exploit data-flow property 2, precompute the output of algorithm 1 for $\frac{P}{N_T}$ cycles and load it in a circular shift register with a programmable feedback tap.

B. Receiver Interface

1) Main Concepts: Data is sampled on both clock edges so that every data item can be safely sampled at least once. The strobe is used to detect on which clock edges data can be safely sampled and when a new data item is available. This is realized by sampling the strobe at two slightly different time instants close to every edge of the clock, and sampling data in the middle of these two time instants. Because the flow of data and the Receiver are both periodic, the information obtained from the analysis of the strobe determines data sampling one or more Periodicity Cycles later.

2) Functionality: To detect if a clock edge could cause metastability, the Receiver samples the strobe on every positive and negative edge of the two clocks clk_R and clk_{R2}, the latter obtained by delaying clk_R by an amount of time that we define as $2T_W$, where T_W is the *time window* of the Receiver. If the two samples match, the clock edge is considered safe for data sampling. Data flow property 3 guarantees that every data item can be safely sampled at least once as long as $\frac{T_H}{2} > 2T_W + t_s + t_h$, where t_s and t_h are respectively the setup and the hold times of the Receiver flipflops. Figure 4 shows a worst-case scenario: data item B cannot be safely sampled on the first falling edge, but data item C cannot arrive before $\frac{T_R}{2} + \frac{T_H}{2}$ and data item B can be safely sampled half a cycle later. In general a data item can be safely sampled more than once. To ensure that the same data is not sampled twice, the strobe sample obtained at t_0 is compared with the strobe sample obtained at $t_0 - \frac{T_R}{2}$: if the two values are different, then a new data item is available at time t_0.

When the two strobe samples obtained at t_0 and $t_0 + 2T_W$ match, it is guaranteed that there was no data/strobe transition

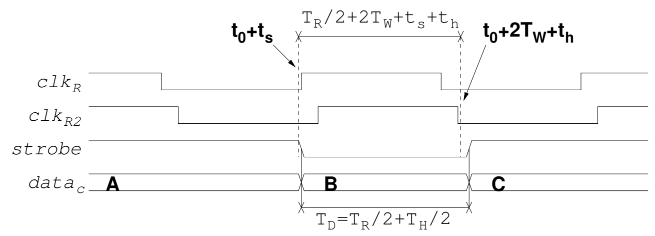

Fig. 4. Strobe sampling: worst-case scenario

between the times $t_0 + t_h$ and $t_0 + 2T_W - t_s$. Data is sampled on every clock edge of clk_{R1}, an additional clock obtained by delaying clk_R by T_W. As long as $T_W > t_s + t_h$, a clock edge considered safe by the strobe analyzer cannot cause the data samplers to encounter metastability.

Since data flow property 2 ensures that both the Receiver and the regulated flow of data are periodic with periodicity PC, the strobe samples obtained at time instants t_0 and $t_0 + 2T_W$ are used to determine data sampling at time $t_0 + T_W + K\,PC$ with K integer.

Figure 5 shows how strobe samples obtained in periodicity cycle PC_i determine data sampling in periodicity cycle PC_{i+1}. On the first rising edge of PC, the two strobe samples match and the sample is accepted. The following strobe transition happens dangerously close to the edge of clk_{R1} and the two samples do not match. The next two clock edges are also safe, but sampling on the last edge of PC would lead to a duplicate. Strobe samples obtained in periodicity cycle PC_i determine data sampling in periodicity cycle PC_{i+1}.

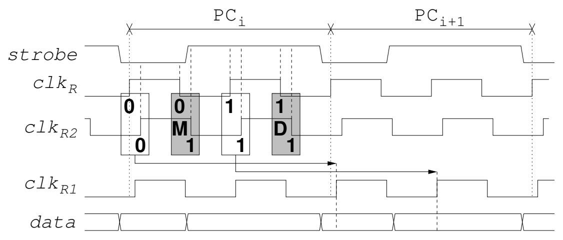

Fig. 5. Data and strobe sampling ($N_T = 2$, $N_R = 3$, $P = 6$)

The limitation $\frac{T_H}{2} > 2T_W + t_s + t_h$ gives the following upper bound for f_H:

$$f_H < \frac{1}{4T_W + 2t_s + 2t_h}$$

If $T_W = T_{W min} = t_s + t_h$:

$$f_H < \frac{1}{6\,(t_s + t_h)}$$

The GRLS Receiver is shown in figure 6.

3) Receiver Registers Stage: As shown in the upper part of figure 6, the Receiver uses the two sets of *input registers* P-REG (positive-edge-triggered) and N-REG (negative-edge-triggered), both clocked by clk_{R1}, to sample the input flow of data. Note that race conditions cannot occur when data is transfered from the sampling registers to the Receiver clock

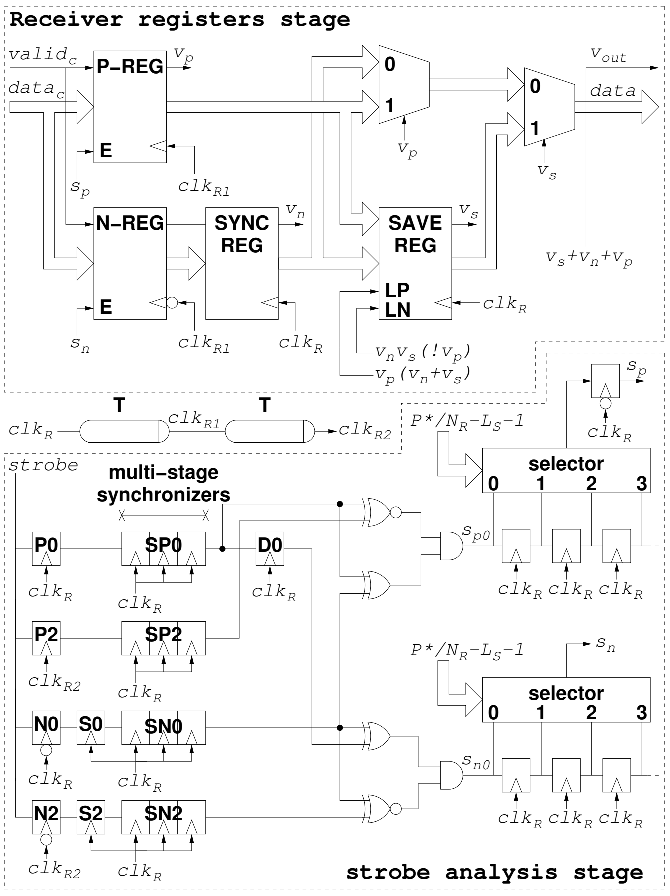

Fig. 6. GRLS Receiver stage

domain as clk_{R1} is delayed compared to clk_R. s_p and s_n are enable signals for P-REG and N-REG. s_p and s_n are generated by the *Strobe Analysis Stage* of the GRLS Receiver and guarantee that neither P-REG nor N-REG will ever become metastable. Note that s_n should arrive before the negative edge of the clock. Signals v_p and v_n indicate validity of the data items contained in P-REG and in the synchronization register SYNC-REG, and are asserted only when the corresponding register is enabled and samples a valid data item from the input.

Register SAVE-REG is used as a single-position FIFO buffer. SAVE-REG is loaded with the output of P-REG when $LP = 1$ and with the output of SYNC-REG when $LN = 1$. When SAVE-REG contains a valid data item, v_s is set to 1. Buffering is necessary because the Receiver can sample up to two data items per clock cycle but can consume only one of them. Since data-flow property 1 guarantees that in K Receiver clock cycles the Receiver will receive at most $K+1$ items of data, a single-element buffer will be sufficient to absorb all bursts in the flow of data even when data is sent at the maximum rate. In other words, data-flow property 1 guarantees that the Receiver will never have to sample two different data elements in the same clock cycle if SAVE-REG already contains a valid data item. The single-stage buffer will also never underflow if the Transmitter sends data at the maximum rate and $f_T \geq f_R$.

Note that the Receiver scheme introduces a slight degradation of the critical path of the system, as two multiplexers are introduced in series with the outputs of the registers and P-REG is clocked by clk_{R1}. If this is an issue, then the designer may want to introduce an additional register at the output of the system. In the remainder of the paper, we assume that no output register is present.

4) Strobe Analysis Stage: The bottom part of the system in figure 6 analyzes the strobe and generates signals s_p and s_n for the upper part of the system (connections are not shown to keep the presentation simple).

The strobe signal is sampled in the Receiver by four samplers: on the positive and negative edges of clk_R by P0 and N0, and on the positive and negative edges of clk_{R2} by P2 and N2.

Strobe samples obtained by N0 and N2 are synchronized to the Receiver clock domain by positive-edge-triggered flipflops S0 and S2. The strobe samplers sample the strobe continuously, and can therefore go metastable. In order to avoid cascaded metastability events, we use high-latency multiple-stage synchronizers SP0, SP2, SN0 and SN2. In traditional synchronized designs such as asynchronous FIFOs, the latency of the synchronizers directly adds latency to the data, and the number of synchronization stages is set as a trade-off between latency and MTBF [19]. Usually a two-stage synchronizer is sufficient to obtain a very high MTBF [20], but in the GRLS Receiver interface multiple-stage synchronizers can be used for additional system safety as they introduce no latency penalty for data.

Flipflop D0 is used to record the value that the output of SP0 had in the past cycle. This is because, at any time, the output of SN0 was sampled half a cycle before the output of SP0, and values for s_p (s_n) are obtained by comparing the outputs of SP0 and SP2 (SN0 and SN2) with strobe samples obtained half a cycle earlier. In particular, s_{p0} (s_{n0}) is set to 1 only when the outputs of SP0 and SP2 (SN0 and SN2) match, which guarantees that strobe transitions happened far enough from the edge of clk_{R1}, and the output of SN0 (D0) is different from the output of SP0 (SN0), which guarantees that a strobe transition happened in the last half cycle so that every data item is sampled only once and as soon as possible.

Programmable-length delay lines are inserted in series to s_{p0} and s_{n0} so that the total delay between strobe sampling and the output of s_p and s_n is PC or a multiple of PC. In particular, s_{p0} and s_{n0} are delayed for a number of cycles equal to $\frac{P^*}{N_R} - L_S - 1$, where L_S is the latency of the synchronizers in terms of clock cycles, and P^* is the smallest multiple of P with $\frac{P^*}{N_R} > L_S$. The delay lines are implemented using a chain of flipflops and a selector. One-hot encoding is used to obtain a simple and fast implementation. Note that the s_p signal should be buffered with a negative-edge-triggered flipflop in order to avoid race conditions as register P-REG is clocked with clk_{R1}.

C. Fast-Receiver, Mesochronous and Static Interfaces

Although the communication method that has been presented can work in all three different communication scenarios, some simplifications can be made if it is known that the system is always fast-Receiver or mesochronous.

In both these scenarios, the single-stage buffer in the Receiver is not needed because the Receiver will never sample two different elements in the same clock cycle. The Receiver stage can be simplified to a system with two registers connected to an output through a multiplexer (figure 7). An additional output register can be inserted if the critical path degradation due to the multiplexer is a problem.

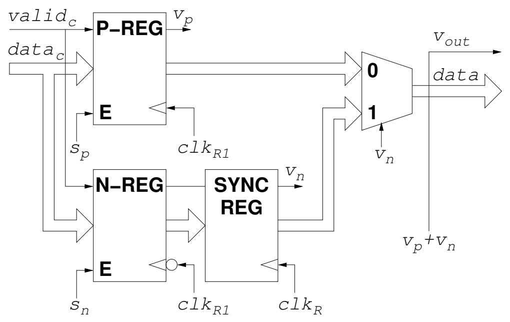

Fig. 7. Receiver registers for mesochronous communication

In a fast-Receiver scenario, an additional simplification can be made, since the minimal time between two successive data outputs by the Transmitter is $T_T \geq T_R + T_H$. As long as $T_H > 2T_W + t_s + t_h$, the Receiver can use a single edge of the clock to sample strobe and data; the system can be simplified to the structure in figure 8. Note however that sampling on both clock edges would increase the upper bound for f_H.

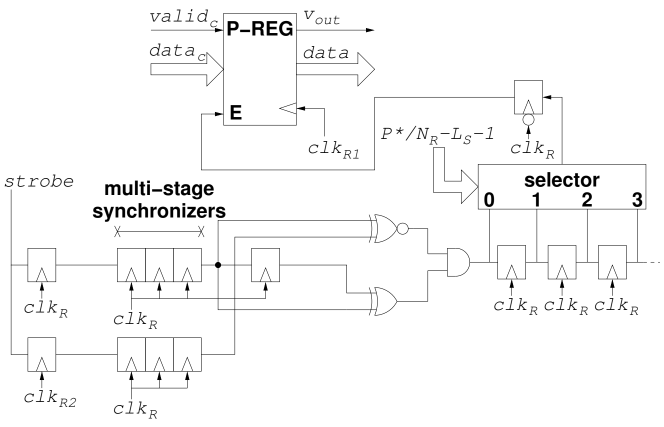

Fig. 8. GRLS Receiver stage for fast-Receiver systems

In certain scenarios it is possible to estimate the skew between the clocks with sufficient accuracy through post-layout static timing analysis. In this case, strobe generation and analysis is not necessary; $\frac{P}{N_R}$ values of s_p and s_n can be precomputed and stored in two programmable-length feedback shift registers; also, registers P-REG and N-REG can be clocked by clk_R and clocks clk_{R1} and clk_{R2} are not necessary.

V. Non-Idealities

A. jitter and misalignments

Until now we considered an ideal scenario with no clock jitter, no data jitter and no misalignments between the strobe and the different data lines. In a real scenario, however, these issues must be considered. Noise in the system introduces jitter, and it is impossible to realize a bundle of data lines with a perfectly-matched delay.

We consider now a more realistic scenario in which each data and strobe line is subject to a jitter $J \leq J_M$, taking into account the clock jitter at the transmitter plus the jitter added by the propagation of the data through the channel. We also assume that the Receiver clock is subject to a jitter $J_{RC} \leq J_{RCM}$. Two data lines have a misalignment $MIS \leq MIS_M$. The GRLS communication scheme can cope with these non-idealities if the delay T_W between clk_R, clk_{R1} and clk_{R2} is increased. As the delay lines are never ideal, we have $T_{Wmin} \leq T_W \leq T_{Wmax}$. By setting

$$\begin{aligned} T_{Wmin} &= t_s + t_h + J_M\,(strobe) + MIS_M \\ &\quad + J_M\,(data) + J_{RCM} \\ &= t_s + t_h + 2J_M + MIS_M + J_{RCM} \end{aligned}$$

we can guarantee that the *sampling strategy* defined by analyzing the strobe will be valid also to sample data. Note that the jitter of the Receiver clock must be accounted for because strobe sampling and data sampling are two events happening in different clock cycles.

In the worst case, data will remain stable on the channel for a time $\frac{T_R}{2} + \frac{T_H}{2} \pm J_M$ and two Receiver samplings will be spaced of $\frac{T_R}{2} \mp J_{RCM}$. In this worst-case scenario, the system will be able to define a sampling strategy as long as:

$$\frac{T_H}{2} - J_M - J_{RCM} > 2T_{Wmax} + t_s + t_h$$

Which leads to the following upper bound for f_H:

$$f_H < \frac{1}{2\,(t_s + t_h + J_M + J_{RCM} + 2T_{Wmax})}$$

With ideal delay lines, assuming $T_W = t_s + t_h + 2J_M + MIS_M + J_{RCM}$, we obtain:

$$f_H < \frac{1}{6\,(t_s + t_h) + 6J_{RCM} + 4MIS_M + 10J_M}$$

See section VII for calculations of a realistic upper bound for f_H in a 90 nm implementation.

B. Clock Drifts

Extending the concept of plesiochronous clocking [2] to rationally-related clock frequencies, we define a plesiorati-ochronous clocking scenario as a scenario in which f_T and f_R are respectively submultiple of two frequencies f_{TH} and f_{RH}, with $f_{TH} \simeq f_{RH} \simeq f_H$ closely matched and differing only for a very small fraction of percentage from a nominal value f_H.

With $f_{TH} = f_{RH} + \Delta f$, the skew between clk_T and clk_R drifts of $\sim \frac{P^* \Delta f}{f_H}$ between strobe sampling and data sampling. The GRLS communication scheme can work in this scenario as long as the timing window T_W is increased to

$$T_W = ts + t_h + 2J_M + MIS_M + J_{RCM} + \frac{P^* \Delta f}{f_H}$$

VI. Comparison With Other Approaches

A. Overhead

The number of flipflops in the GRLS Receiver interface is given by:

$$N = 4W + 7 + 4S + 2\left(N_{max} - S - 1\right)$$

where W is the width of the data lines, S is the number of synchronization stages, and N_{max} is the maximum value for N_T and N_R. The Receiver requires 4 flipflops for every data line, which corresponds to the overhead of standard mesochronous synchronizers [5]–[7]. Our method, however, is more flexible as it supports rationally-related clock frequencies. When the system is simplified for the mesochronous case, the overhead of the GRLS Receiver is reduced to 3 flipflops per data line.

Perhaps the most straightforward solution for ratiochronous clock-domain crossing consists in introducing in the channel an additional unit clocked by a clock clk_H running at frequency f_H, connected with the Transmitter and the Receiver using upsampling mesochronous links. The ideal situation consists in having no skew between the high-frequency unit and the Receiver clock, as this would allow the high-frequency unit and the Receiver to communicate synchronously and would require a single mesochronous link. However, this solution is unpractical because in a modern chip it would be difficult to generate a high-frequency clock synchronous with the Receiver clock: clk_R is obtained from a clock running at f_H, but clk_R is delayed by the local clock tree before reaching the leaf cells of the Receiver. The only available solution consists in using two separate mesochronous links, which would require at least 8 flipflops per data line and would add a significant latency to the system (see section VI-B). Also, the double-mesochronous solution would introduce a considerable power overhead compared to the GRLS scheme as it would require the introduction of a unit clocked with f_H. The overhead would be particularly high for systems in which f_H is significantly higher than f_T and f_R. Besides, our scheme can work also when no physical frequency f_H is present on the chip, such as in certain plesioratiochronous systems. Using the approaches in [16] and [17] would not improve the overhead figures.

B. Throughput and Latency

The system is an optimal-throughput communication scheme, as communication can always happen at the maximum possible rate of one data item for every clock cycle of the slowest of the two units.

To study the latency introduced by the communication scheme, let us consider a condition in which all buffers are empty and a single data item is sent from the Transmitter to the Receiver. The latency of an item of data is defined as the difference between the time at which the data item is consumed by the Receiver and the time at which the same data item was produced in the Transmitter. The latency is given by the sum of the propagation delay through the channel T_C plus the latency L of the communication scheme. Ignoring setup and hold times, if no regulation is present, data is immediately output and $L \leq T_R$. Note that this latency is optimal, as data can arrive at any time at the Receiver but can be consumed only on positive clock edges.

For regulated fast-transmitter flows, given N_T and N_R, it is possible to calculate an upper bound on the latency introduced by the system. The maximal time between two successive data outputs is, based on data-flow property 4, equal to $T_T \left\lceil \frac{N_R}{N_T} \right\rceil$, and corresponds to the maximal delay introduced by the regulator. Remembering that $T_T = T_R \frac{N_T}{N_R}$ and adding one additional factor T_R to account for the fact that data can arrive at the Receiver at any time, we have:

$$L \leq T_R \left(1 + \frac{N_T}{N_R} \left\lceil \frac{N_R}{N_T} \right\rceil \right)$$

The average latency of our communication system for different combinations of N_R and N_T can be calculated by simulation. Since the latency introduced by the system depends on the skew between the clocks and the channel delay, values obtained from a set of different simulations are averaged. In table I are reported latency values for different combinations of N_T and N_R.

The table reports worst-case and average-case latency values for two different situations: when there is skew between the clocks and when there is no skew. The no-skew figures are reported to compare with Sarmenta's Rational Clocking approach from [14]. The lazy algorithm of Rational Clocking leads to worse latency figures for fast-Transmitter systems compared to the GRLS interface.

Supposing two synchronization stages and taking as a reference the implementation in [19], the worst-case and average latencies of asynchronous FIFOs are respectively $3T_R$ and $2.5T_R$, which are always higher than the figures of our communication system.

It can also be noted that, since our method samples data at the Receiver on both clock edges, if the input stage of the system allows a timing budget of half a clock cycle, it is possible to delete the synchronization register SYNC-REG in figure 6 to reduce the overhead and subtract $\frac{T_R}{2}$ to all average and worst-case latency figures. Data items sampled on the negative edge of the clock become then immediately available at the output.

State-of-the-art mesochronous links implemented such as in [5], assuming that 4-stage FIFOs are used, introduce a worst-case latency of $3T_R$. A double-mesochronous solution for rational clocking using a high-frequency unit in the chan-

$N_T : N_R$	Worst-case latency (T_R)			Average latency (T_R)		
	Skew GRLS	No skew GRLS	RC	Skew GRLS	No skew GRLS	RC
$N_R \leq N_T$ (unreg.)	1			0.5		
$N_T = \frac{N_T}{K}$	2			$1 + \frac{1}{2K}$		
2 : 3	2.333	1.667	2.000	1.389	1.333	1.667
2 : 5	2.200	1.800	2.000	1.220	1.400	1.600
3 : 5	2.200	1.800	2.800	1.340	1.400	1.800
3 : 7	2.286	1.857	2.714	1.234	1.429	1.571
5 : 7	2.429	1.857	2.429	1.418	1.429	1.571
8 : 11	2.455	1.909	2.000	1.426	1.455	1.545
7 : 12	2.167	1.917	2.750	1.326	1.458	1.625
12 : 17	2.412	1.941	2.000	1.414	1.471	1.529
16 : 17	2.882	1.941	2.000	1.497	1.471	1.529

TABLE I
LATENCY ANALYSIS AND COMPARISON WITH THE RATIONAL CLOCKING (RC) APPROACH

nel would thus introduce a worst-case latency of at least $3T_R + 3T_H$, which is always higher than the latency introduced by our interface.

VII. IMPLEMENTATION RESULTS

In TSMC 90 nm CMOS technology, we have $t_s + t_h = 31ps$. Based on the equations in section V, a realistic value for T_{Wmin}, assuming a medium-length interconnect with jitters and misalignments of $20\,ps$, is $111\,ps$. A delay line is built using 10 inverters in the technology library. The delay line introduces a delay of $112\,ps$ in the best case and $238\,ps$ in the worst case. In turn, this leads to the upper bound $f_H < 910\,MHz$.

The delay lines have both an area of 16 Gate Equivalents (GEs). In table II are reported the area overheads of the communication system in terms of GEs. Synthesis has been done in TSMC 90 nm technology targeting a working frequency of $1\,GHz$. The parameter N_{max} determines the maximal value for N_T and N_R. For the fast-Transmitter scenario, the regulator was realized using a feedback shift register for $N_{max} = 16$ and an FSM implementing algorithm 1 for $N_{max} = 32$ as these were the optimal-area solutions.

Scenario	Regulator	Strobe analysis	Receiver registers	Total
fast-Tran ($N_{max} = 16$)	165	293	1450	1922
fast-Tran ($N_{max} = 32$)	229	513	1450	2205
fast-Rec ($N_{max} = 16$)	-	150	274	438
fast-Rec ($N_{max} = 32$)	-	260	274	506
mesochronous	-	141	1082	1237

TABLE II
AREA OVERHEADS OF THE COMMUNICATION SYSTEM IN GATE EQUIVALENTS

VIII. CONCLUSION

In conclusion, we have shown how the overhead and performances of our interface are close to those of state-of-the-art mesochronous interfaces, despite allowing greater flexibility. Using our approach, a GMLS system can be turned into a much more flexible GRLS system introducing only a small overhead. We have also shown latency, overhead, flexibility and complexity advantages over all other approaches that have so far been proposed to cross the boundary between clock domains with rationally-related clock frequencies.

We now plan to further validate our communication scheme by implementing a complete AXI-based GRLS chip using our interfaces for communication.

ACKNOWLEDGMENTS

This research is supported and financed by the EU MOSART project (grant agreement no. 215244).

The authors acknowledge Cadence for the support received through the Cadence Academic Network.

The authors would like to thank Johnny Öberg for his helpful advice.

REFERENCES

[1] International Technology Roadmap for Semiconductors Report, 2007
[2] P. Teehan et al., ”A Survey and Taxonomy of GALS Design Styles,” IEEE Design & Test of Computers, vol.24, no.5, Sept.-Oct. 2007
[3] A. Hemani et al., ”Lowering Power Consumption in Clock by Using Globally Asynchronous Locally Synchronous Design Style,” DAC 1999
[4] D. G. Messerschmitt, ”Synchronization in Digital System Design,” IEEE Journal of Selected Areas in Communications, vol. 8, no. 8, Oct. 1990
[5] M. R. Greenstreet, ”Implementing a STARI Chip,” ICCD 1995
[6] S. R. Vangal et al., ”An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS,” IEEE Journal of Solid-State Circuits, vol.43, no.1, Jan. 2008
[7] E. Nilsson and J. Oberg, ”Reducing Power and Latency in 2-D mesh NoCs Using Globally Pseudochronous Locally Synchronous Clocking,” CODES + ISSS 2004
[8] I. Loi et al., ”Developing Mesochronous Synchronizers to Enable 3D NoCs,” DATE 2008
[9] D. M. Chapiro, ”Globally Asynchronous Locally Synchronous Systems,” PhD thesis, Stanford University, Oct. 1984
[10] K. Y. Yun and R. P. Donohue, ”Pausible Clocking: a First Step Toward Heterogeneous Systems,” ICCD 1996
[11] D. Kim et al., ”Asynchronous FIFO Interfaces for GALS On-Chip Switched Networks,” International SoC Design Conference, 2005.
[12] S. Herbert and D. Marculescu, ”Analysis of Dynamic Voltage/Frequency Scaling in Chip-Multiprocessors,” ISLPED 2007
[13] T. Olsson et al., ”A Digitally Controlled on-Chip Clock Multiplier for Globally Asynchronous Locally Synchronous Systems,” Midwest Symposium on Circuits and Systems, 1999
[14] L. F. G. Sarmenta et al., ”Rational Clocking,” ICCD 1995
[15] J. Mekie et al., ”Interface Design for Rationally Clocked GALS Systems,” ASYNC 2006
[16] F. Mu and C. Svensson, ”Self-tested self-synchronization Circuit for Mesochronous Clocking,” IEEE Transactions on Circuits and Systems, vol.48, no.2, Feb. 2001
[17] F. Vitullo et al., ”A Mesochronous Physical Link Architecture for Network-on-Chip Interconnects,” PRIME 2007
[18] A. Chakraborty and M. R. Greenstreet, ”Efficient Self-Timed Interfaces for Crossing Clock Domains,” ASYNC 2003
[19] C. E. Cummings and P. Alfke, ”Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons,” Synopsys Users Group Conference, 2002
[20] D. J. Kinniment et al., ”Synchronization Circuit Performance,” IEEE Journal of Solid-State Circuits, vol. 37, no. 2, Feb. 2002