A Low-Cost High-Speed Source-Synchronous Interconnection Technique for GALS Chip Multiprocessors

Anh T. Tran, Dean N. Truong, and Bevan M. Baas
University of California - Davis
{*anhtr, hottruong, bbaas*}*@ucdavis.edu*

***Abstract*—The globally asynchronous locally synchronous (GALS) design style for a large area chip has become increasingly attractive due to the difficulty of designing global clocking circuits at high clock frequencies in the GHz range. In this paper, we present a high-speed interconnect network for a GALS multiprocessing system composed of a 2-D mesh array of processors. Processors are locally clocked by their own oscillators and communicate together using a static circuit-switched technique combined with a source-synchronous communication scheme. A technique to maximize the timing reliability on long-distance interconnects at high clock rates is proposed that is area and power efficient with low latency and allows a sustained ideal peak throughput of one word per cycle.**

I. Introduction

In deep submicron CMOS, the parasitic effects of wire interconnects—resistive, capacitive, and inductive—on system performance are no longer negligible. Clocking circuits have become increasingly difficult to design for large chip sizes and at high clock rates. Additionally, high speed global clock trees continue to consume a significant portion of the power budget. At worst, its power consumption can be up to 40% of the total power dissipation of a whole system [1]. To minimize these issues, Globally Asynchronous Locally Synchronous (GALS) architectures have shown promise [2]. With GALS, a large chip architecture is partitioned into multiple frequency domains with each domain clocked synchronously while inter-domain communication is achieved asynchronously.

There are two well-known techniques for inter-domain communication: clockless handshaking and source-synchronous. Handshaking technique uses multiple phases of exchanging *request/confirm/valid/ack* signals to transfer data across clock domains. Unfortunately, its round-trip latency is high. Besides that, asynchronous handshaking circuits have complex control circuits which are difficult to verify in traditional CAD flows, and also demand a comparatively larger area and energy requirement [3], [4]. On the other hand, source synchronous communication architectures only need a source clock signal to be sent along with the data to the destination. In this method, a dual-clock FIFO at the destination is used to buffer and synchronize data between two clock domains. This method achieves high efficiency by obtaining a peak throughput of one data word per cycle with lower power consumption [5], [6].

A source synchronous communication link can also be scaled, allowing long-distance interconnection between two arbitrary processors over a large many-core chip. However, at high clock rates, stability and reliability through long interconnect wires is difficult to achieve [7]. One low cost solution is by pipelining long-distance wires using several intermediate registers [8]. However, without careful design considerations, this method is likely to encounter metastability due to timing violations along the interconnect link.

In this paper, we investigate the timing requirements for successful data transfer using a source synchronous network on a GALS many-core platform. After considering the various methods for ensuring and/or increasing reliable data transfer, we propose a solution which achieves increased communication robustness at a low area cost. The outline of this paper is as follows: Section II discusses the causes of timing violations during a source synchronous transfer of data. Then we analyze current possible solutions for avoiding violations in Section III. Our proposed low cost solution is presented in Section IV. Section V quantitatively evaluates solutions and, finally, Section VI concludes the paper.

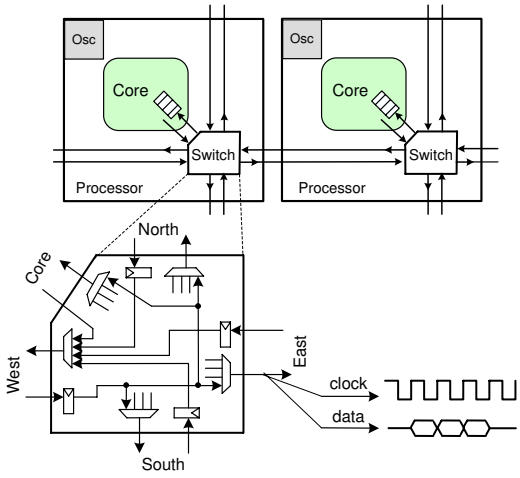

Fig. 1. Circuit-switched interconnect architecture for a processor simplified to highlight communication along the West port

II. Source-synchronous Interconnect Architecture for GALS Chip Multiprocessors and Its Timing Issue

We will consider the issue of reliable source synchronous communication at high clock frequencies in the GHz range on a GALS many-core computational platform comprised of an array of processors; each processor is clocked by its own local oscillator. Processors are interconnected by a 2-D mesh circuit-switched network supporting source synchronous communication [9].

Since each processor has its own clock domain, data between two processors are transferred asynchronously. Due to its advantages over clockless communication as briefly explained in Section I, source synchronous communication is adopted by our platform with pipelined long-distance interconnect links [8].

Figure 1 depicts the interface between two neighboring processors in the platform. The switch inside each processor has five ports: the Core port which is connected to its local core, and the North, South, West, and East ports which are connected to its four nearest neighbors. As shown in the figure, an input from the West port can be configured to go out to any port among the Core, North, South, East ports and vice versa. For simplicity, Fig. 1 only shows full connections to and from the West port; all the other ports are connected in a similar fashion.

The multiplexers of the switch are pre-configured before runtime which fixes the communication link among the processors. Thus,

978-1-4244-3828-0/09/$25.00 ©2009 IEEE

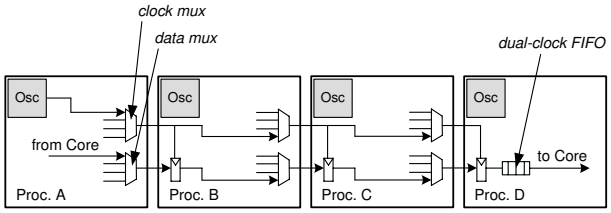

Fig. 2. Long-distance source synchronous communication through two intermediate processors

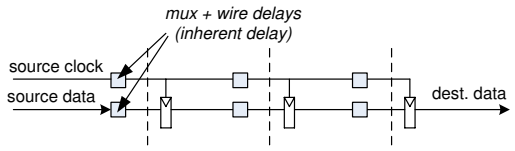

Fig. 3. A simplified illustration of the communication link of Fig. 2

the circuit-switched link is guaranteed to be independent and never shared. So long as the destination processor's FIFO is not full, a one data word per cycle throughput can be sustained. This compares favorably to a packet-switched network whose runtime network congestion can significantly degrade communication performance [6]. Our interconnect architecture is well suited for DSP applications with fixed interconnect requirements among processors.

Figure 2 shows an example of a communication link that is configured to connect two long-distance processors. This link passes through two intermediate processors, Proc. B and Proc. C, which are in between the source and destination processors, Proc. A and Proc. D, respectively. The figure also shows both clock and data being multiplexed by the circuit-switched architecture. The destination processor (Proc. D) uses a dual-clock FIFO to buffer the received data before processing. Its FIFO's write port is clocked by the source clock of Proc. A, while its read port is clocked by its own oscillator, and thus supporting GALS communication [10].

Figure 3 is a simplified version of Fig. 2 focusing only on the impact of delay on source synchronous timing. The dotted lines represent the boundary between two nearest processors. In order to meet timing, the clock and data links should have nearly equal wire and multiplexer (i.e. gate) delays. These delays are combined and denoted as an *inherent delay*. We can constrain the difference between the inherent delays of the clock and data links to be small in the synthesis and the place and route phases of the backend implementation.

Data and clock are sent by the source processor to the destination processor through a sequence of intermediate multiplexers and registers, and each data word is valid for one cycle. All registers are latched by the rising clock edges. Because the delay of the clock and data is generally close, a timing violation can occur as illustrated by the waveform in Fig. 4. Also, the data bus can have mismatches due to variations and crosstalk, and the clock can have jitter that

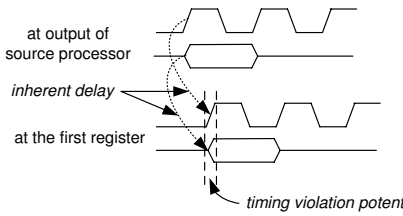

Fig. 4. Timing waveforms of source synchronous communication from the source to the next register

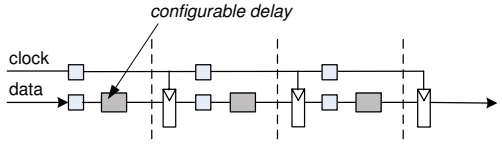

Fig. 5. Configurable delays on the data bus of a long-distance link

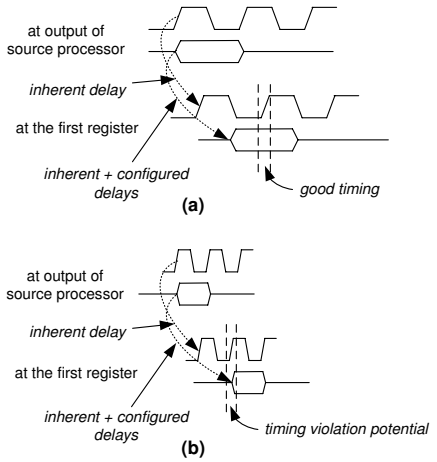

Fig. 6. Timing waveform after adding configurable delays on the data bus. (a) Correct timing when delay is appropriately set corresponding to a specific clock frequency. (b) Timing violation when changing frequency while retaining the previous setting.

causes unreliable communication in actual chip implementations. This issue is more pronounced when several data words are sent in the consecutive clock cycles. In the next sections, we will analyze possible solutions used to mitigate or eliminate the issue with source synchronous on-chip communication, and then present our simple yet novel low cost solution.

III. Related Work

A. Configurable Delay Method

An intuitive way to avoid the timing violation presented in the previous section is to purposefully add delay to the data or clock signal such that the rising edge is sufficiently far from the region where the data bits change. This ensures that the data is stable within the setup and hold time window of a master-slave latch or D flip-flop [5]. Fig. 5 shows a long-distance interconnect architecture where a configurable delay circuit is added along the data bus at each processor. The basic idea is that we delay the data so as to latch it correctly at the following register on the next rising clock edge. Depending on experimental testing of actual silicon, the delay is configured for every link in order for the following setup time and hold time constraints

$$D + t_{clk-q} + t_{setup} < T \quad (1)$$

$$t_{hold} < D + t_{clk-q} \quad (2)$$

to be satisfied. Here, D is the configured delay; t_{clk-q}, t_{setup} and t_{hold} are the clock-to-output delay, setup time and hold time of the register, respectively; and T is the source clock cycle time.

Figure 6(a) illustrates the case when the configurable delay is set appropriately in order to meet both constraints. Equation (2) shows that when satisfying the hold time, D is independent of the source clock period. However, after we fix a value for D, setup time violation will occur once the frequency increases to a point where T is small enough to make Eq. (1) false as pictured in Fig. 6(b). This situation

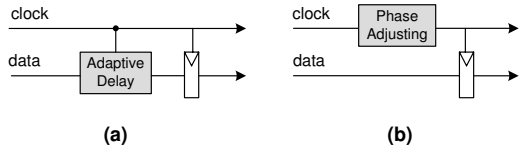

Fig. 7. (a) Adaptive delay of the data bus through clock signal capturing. (b) Adaptive clock phase matching using through self correction.

is more serious for a GALS multiprocessing system that utilizes Dynamic Voltage and Frequency Scaling (DVFS).

For high speed digital communication, both constraints become even harder to satisfy when we further consider VLSI issues in sub-micron technology such as clock jitter, data crosstalk, and variations.

When considering clock jitter, in the worst-case, the clock cycle time can be as small as $T - 2t_{jitter}$ [11], so Eq. (1) becomes:

$$T > D + t_{clk-q} + t_{setup} + 2t_{jitter} \tag{3}$$

When further considering the effects of variations and crosstalk, D, t_{clk-q}, t_{setup}, and t_{hold} all become probabilistic variables that makes the range of safe clock frequencies more limited. This type of analysis will be left to future work.

B. Other Methods

To deal with a dynamic source clock frequency, a method is by adjusting the delay of the data bus adaptively corresponding to clock frequency in order for the Eq. (3) is always satisfied (Fig .7(a)) [12].

Other methods use clock recovery schemes, typically found in off-chip applications (e.g. PCI Express). So far PLLs and delay lines have been the most popularly implemented solutions [13], [14]. These circuits readjust the phase of the clock (i.e. delay of the clock) to only latch at the stable region of data (see Fig .7(b)). They require feedback circuits that often contain complicated processing blocks, which have much greater difficulty when designed to operate on high frequency systems.

Although these adaptive solutions largely guarantee correct data transfer, the dynamic nature requires complicated circuitry with higher power consumption, larger area, and greater verification complexity. Additionally, as shown in Fig. 6(a), the latency added on each intermediate processor by these methods (or by the configurable delay solution) is T plus an *inherent delay*.

Another method to deal with metastability is through the use of a synchronizer consisting of multiple registers on the data bus instead of single register for each intermediate processor [15]. Again, this method has high area cost and latency.

IV. Proposed Solution

A. Initial Observations

In Section II we have observed that naïvely implementing a source synchronous communication architecture results in potential timing violations (Fig. 4). Furthermore, static configurable delay along the data or clock links only mitigates the problem while requiring that the designer meet timing constraints that are unpredictable until tape-out and testing.

In order to develop a better solution without resorting to costly adaptive circuits, we note that the data word's stable region at the first register will always lie at the falling edge of the source clock signal. This is illustrated in Fig. 8 with an assumption of a 50% duty cycle clock. With the same observation, the second register should be latched by a following rising edge as seen in the figure. Inevitably, this method requires that all registers along the interconnect link be clocked by either rising or falling clock edges in an alternating fashion

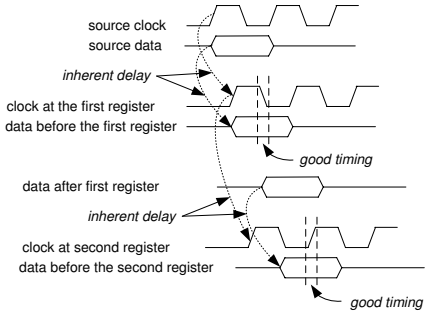

Fig. 8. Timing using alternating clock edges along two registers

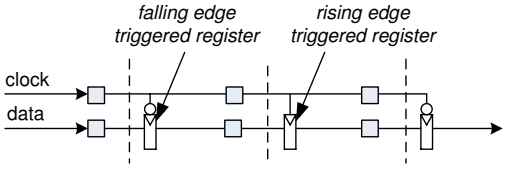

Fig. 9. Alternating rising and falling edge-triggered registers along the long-distance link

as depicted in Fig. 9. To make this method operate correctly, we must satisfy the following timing constraints:

$$t_{clk-q} + t_{setup} < \frac{T}{2} \tag{4}$$

$$\frac{T}{2} + t_{hold} < T + t_{clk-q}. \tag{5}$$

When these are combined while considering jitter, these constraints become:

$$T > 2 \cdot max\{t_{setup} + t_{clk-q},\ t_{hold} - t_{clk-q}\} + 2t_{jitter}. \tag{6}$$

So long as setup time, hold time, jitter, and clock-to-output delay are small, the interconnect will be able to reliably operate at very high clock frequencies.

A side benefit when compared with the previous solutions is that the latency between neighboring processors is now just $T/2$ plus an *inherent delay* instead of T plus an *inherent delay* when using the configurable/adaptive delay methods, or even multiple cycles when using synchronizer circuits.

To implement this method, however, there are two issues that we must consider. First, since rising and falling edge-triggered registers alternate along the interconnect, it appears that we will have to design two versions of communication links, which adds more difficulties for the backend design. Second, a processor may be configured to be a source, a destination or an intermediate processor depending on the application(s) mapped, so it should support both types of edge-triggered registers.

Figure 10 shows a possible design to make all the links identical while supporting both edge-triggered registers. Multiplexers are pre-configured before runtime to establish a long-distance interconnect set with the correct alternation of rising and falling edge-triggered registers. Unfortunately, this method is area inefficient requiring more multiplexers and registers along the data bus.

B. A Low Cost Solution

From the initial idea above, we derive a simple solution by placing an inverter along the clock path before each register as shown in Fig. 11. This ensures that no matter which processor is a source, destination, or intermediate, the clock edges always alternate correctly at all the registers. Since the inverter adds a delay to the clock path, the data path should have near identical buffers in terms of delay along the data bus to match their delays as seen in Fig. 12.

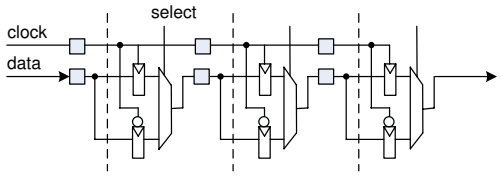

Fig. 10. Alternating edge-triggered registers using configuration along a long-distance link

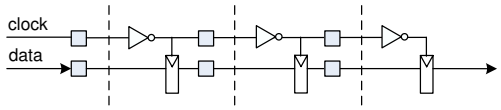

Fig. 11. Adding an inverter before each rising edge-triggered register along the clock path

V. Evaluation

To quantitatively evaluate the effectiveness of the various approaches, we modeled and simulated a static C^2MOS edge-triggered register [16] at the conventional operating condition using HSPICE. Table I lists the simulation result of average setup time, hold time, clk-to-output delay and one FO4 gate delay corresponding to some CMOS nodes specified by the Predictive Technology Model [17].

For the configurable delay method described in Section III-A, to achieve the safest margins for satisfying hold time, D is normally set between 5 FO4 and 10 FO4 [5]. Fig. 13(a) shows the maximum frequency curves over several technology processes (ignoring clock jitter). Our solution can sustain a very high clock frequency that is about 4.6 GHz in 65 nm and 6.8 GHz in 45 nm technology. If we assume that clock jitter is approximately 10% of the total clock cycle, an interconnect link can run at a max frequency of about 3.7 GHz in 65 nm and 5.4 GHz in 45 nm.

This solution also has a much lower communication latency compared to the configurable delay method illustrated in Fig. 13(b). Since the latency of other methods are equal to or larger than that of the configurable delay method, our proposed solution clearly has the lowest communication latency regardless of the distance between the source and destination. Moreover, our solution is very simple without complicated circuits, therefore, it invariably will consume less power and area than known alternative methods.

VI. Conclusion

In this paper, we have just presented a circuit-switched interconnect network for GALS chip multiprocessors. The interconnect links of the network use source synchronous clocking to sustain a peak throughput of one word per cycle. Our proposed solution for dealing with the inherent timing problems on these high-speed pipelined interconnects offers a more robust method to meet source synchronous timing requirements in a simple yet effective manner.

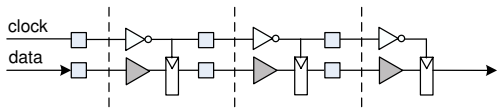

Fig. 12. Adding buffers on data links to balance the delay with inverters on clock links

TABLE I
Simulated Timing

Technology (nm)	clk-to-output (ps)	setup time (ps)	hold time (ps)	FO4 (ps)
130	142	28	29	34
90	121	21	23	26
65	89	19	17	19
45	58	15	11	12

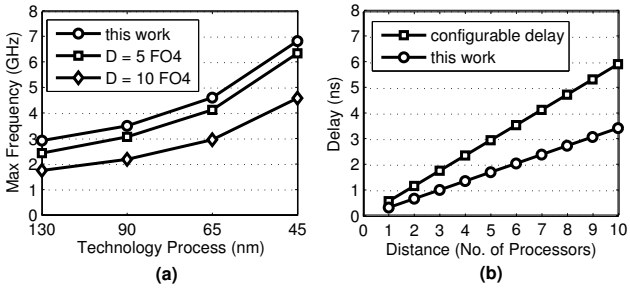

Fig. 13. (a) Maximum frequency sustained corresponding to the technology. (b) Communication latency versus the distance between two processors with clock frequency of 2 GHz and an inherent delay of 5 FO4 in 65 nm.

This allows us to more safely guarantee successful data transfers at high clock frequencies while under the real-world uncertainties that future high-speed communication on submicron technologies will inevitably incur.

Acknowledgments

This work was supported by IntellaSys, a VEF Fellowship, SRC GRC Grant 1598 and CSR Grant 1659, ST Microelectronics, UC Micro, NSF Grant 0430090 and CAREER Award 0546907, Intel, and SEM.

References

[1] V. Tiwari, D. Singh, et al., "Reducing power in high-performance microprocessors," in *Design Automation Conference (DAC)*, June 1998, pp. 732–737.

[2] M. Krstić, E. Grass, et al., "Globally asynchronous, locally synchronous circuits: Overview and outlook," *IEEE Design and Test of Computers*, vol. 24, no. 5, pp. 430–441, Sept. 2007.

[3] B. R. Quinton, M. R. Greenstreet, and S. J.E. Wilton, "Asynchronous ic interconnect network design and implementation using a standard ASIC flow," in *IEEE International Conference of Computer Design (ICCD)*, Oct. 2005, pp. 267–274.

[4] E. Beigne and P. Vivet, "Design of on-chip and off-chip interfaces for a GALS NoC architecture," in *IEEE Intl. Symposium on Asynchronous Circuits and Systems (ASYNC)*, Mar. 2006.

[5] Z. Yu and B. M. Baas, "Implementing tile-based chip multiprocessors with GALS clocking styles," in *IEEE International Conference of Computer Design (ICCD)*, Oct. 2006.

[6] Y. Hoskote, S. Vangal, et al., "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sept. 2007.

[7] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE*, pp. 490–504, Apr. 2001.

[8] Z. Yu and B. M. Baas, "Low-area interconnect architecture for chip multiprocessors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2008, pp. 2857–2860.

[9] D. Truong, W. Cheng, et al., "A 167-processor 65 nm computational platform with per-processor dynamic supply voltage and dynamic clock frequency scaling," in *Symposium on VLSI Circuits*, June 2008.

[10] R. Apperson, Z. Yu, et al., "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," *IEEE TVLSI*, vol. 15, no. 10, pp. 1125–1134, Oct. 2007.

[11] J. M. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, chapter 10, Prentice-Hall, New Jersey, U.S.A, second edition, 2003.

[12] M. E. S Elrabaa, "An all-digital clock frequency capturing circuitry for NRZ data communications," in *IEEE Intl. Conference on Electronics, Circuits, and Systems*, Aug. 2006, pp. 106–109.

[13] M. Kihara, S. Ono, and P. Eskelinen, *Digital Clocks for Synchronization and Communications*, Artech House, Inc, Norwood, MA, U.S.A, 2003.

[14] S. W. Moore, G. S. Taylor, et al., "Self calibrating clocks for globally asynchronous locally synchronous systems," in *IEEE Intl. Conf. on Computer Design (ICCD)*, Sept. 2000, pp. 73–78.

[15] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*, Cambridge University Press, New York, U.S.A, 1998.

[16] D. Marković, B. Nikolić, and R. Brodersen, "Analysis and design of low-energy flip-flops," in *IEEE ISLPED*, Aug. 2001, pp. 52–55.

[17] W. Zhao and Y. Cao, "Predictive technology model for nano-CMOS design exploration," *ACM JETC*, vol. 3, no. 1, pp. 1–17, Apr. 2007.