A Synchronous Interface for SoCs with Multiple Clock Domains

Visvesh Sathe, Conrad Ziesler*, Marios Papaefthymiou*,*
Suhwan Kim† and Stephen Kosonocky‡

*EECS Department
U. Michigan
Ann Arbor, MI 48109, USA
vssathe,cziesler,marios@eecs.umich.edu

†EECS Department
Seoul National U.
Seoul 151-744, Korea
suhwan@ee.snu.ac.kr

‡Low-Power Digital Circuits
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
stevekos@us.ibm.com

Abstract

This paper presents an interface technique for hazard-free communication among synchronous intellectual property (IP) cores that belong to different clock domains. By allowing each IP core to run at its most efficient operating point, the proposed interface can yield significant power reductions in synchronous System-on-Chip (SoC) designs. The technique is provably correct and achieves maximum throughput when transferring data across long distances. Its performance has been assessed through Hspice simulations.

1. INTRODUCTION

In SoC designs with multiple IP cores, the use of a single operating frequency may result in excessive power dissipation, as individual cores may operate faster than required by their workloads. However, the deployment of multiple frequencies requires a robust and efficient inter-core communication methodology that is immune to synchronization failure when crossing clock domains.

Previously proposed approaches for crossing clock domains have addressed the general problem with arbitrary clock frequencies [1, 2, 3, 4, 5], resulting in reduced throughput, especially over long-latency channels. In many SoCs, however, clock domains are generated from a master clock using dividers. This fact can be exploited to enable the efficient and provably correct inter-core communication across multiple clock domains with different frequencies.

This paper presents an interface technology, called Direct Conversion Method (DCM), that is specifically aimed at SoCs with multiple clock domains derived by dividing a master clock. To allow for hazard-free communication, DCM uses a control system to directly "synchronize" the communicating cores without using handshaking protocols. DCM provides several key advantages over existing interface techniques:

- Provably zero probability of synchronization failure.
- Minimal communication latency in the case of inter-core communication across long distances.
- Maximum communication throughput, regardless of the physical separation of the two cores.
- Straightforward interface implementation based on widely used circuit techniques.

The correct operation of DCM has been proven analytically. The efficiency and performance of DCM has been assessed through HSPICE simulations of two cores in a $0.13\mu m$ logic process running at 500MHz and 333.3MHz, respectively.

2. INTERFACE OVERVIEW

DCM is applicable to SoCs with clock domains derived from a master clock frequency through integer division. Since the ratio of the resulting clock periods is rational, if the latching edges of two clock domains are aligned at some time, they are guaranteed to periodically align thereafter. Furthermore, it can be shown that the time elapsed between a sender clock edge and a non-coinciding receiver clock edge is at least one full time period of the master clock. Thus, synchronization failure can be completely avoided, without any handshake circuits between the clock-domains, if simple timing constraints are met.

Periodic alignment of two coincident clock edges can be enforced by dynamically tuning a delay line. The control circuitry used to perform this tuning in DCM, called Periodic Clock Alignment (PCA) module, divides down the clock from both clock domains to their "least-common multiple" period and tunes the delay line based on the phase difference between these derived clocks. This derivation is always possible due to the rational relation of the two clock periods.

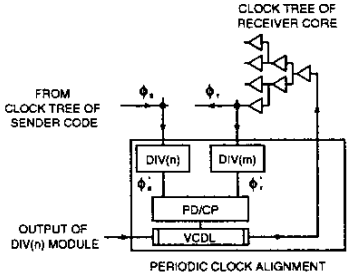

Figure 1: Dynamic tuning for periodic clock alignment

Figure 1 shows the PCA module. The sender clock ϕ_s and receiver clock ϕ_r have clock periods $m \cdot T$ and $n \cdot T$, respectively, where T is the period of the master clock, and m and n are co-prime. (The extension to n and m that are not co-prime is straightforward.) The PCA consists of a conventional Delay Locked Loop (DLL) and two counters that divide ϕ_s and ϕ_r by n and m respectively, so that the periods of both clocks, ϕ_s' and ϕ_r', are $m \cdot n \cdot T$. The remaining modules, i.e., the Phase Detector (PD), the Charge Pump (CP) and the Voltage-Controlled Delay Line (VCDL), are all standard modules of a conventional DLL. The PCA module uses the signals ϕ_s' and ϕ_r' to tune the delay on ϕ_r to ensure that the clock edges of ϕ_s and ϕ_r are edge-aligned periodically with period $m \cdot n \cdot T$.

Figure 2 shows DCM for a slow sender and a fast receiver. Data produced on the latching edge of the sender's clock is latched by the latching edge of the receiver immediately thereafter. The

0-7803-8445-8/04/$20.00 ©2004 IEEE

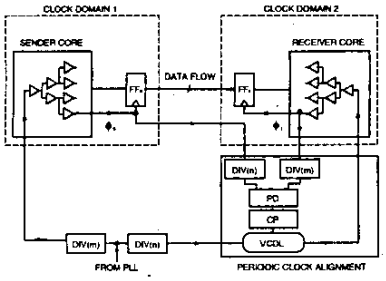

Figure 2: DCM for adjacent cores with $T(\phi_s) > T(\phi_r)$

cores are directly connect to each other, and data transfer takes place directly without any handshake or communication protocol. The clock signal is routed along with the data from the sender to the receiver, providing the necessary timing information to enable hazard-free latching by the receiver.

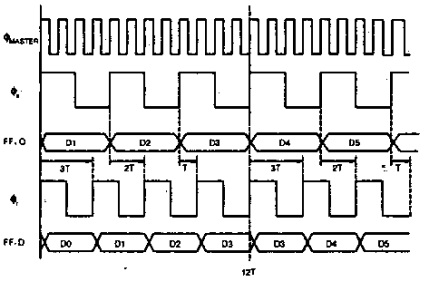

Figure 3: Data flow across clock domains

The timing of the transmitted data is shown in Figure 3. The time difference between a write edge and the next read edge is at least T. Observe that D3 is latched twice by the receiver. Such errors due to duplicate reads can be avoided using a toggle bit that flips every time new data is sent.

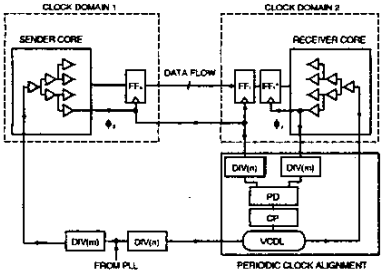

Figure 4: DCM over long distances with $T(\phi_s) < T(\phi_r)$

Figure 4 describes DCM when cores are at an appreciable distance from each other. Latch FF_r is clocked by the sender's clock that is routed to the receiver. If interconnect latency between FF_s and FF_r exceeds $m \cdot T$, then additional latches can be inserted to pipeline the interconnect, allowing DCM to sustain full throughput. The sender clock signal is used to drive the clock pins of these pipeline latches. Regardless of interconnect length, it must be ensured that the last latch clocked by the sender clock domain is adjacent to FF_r*. The adjacency condition ensures that the data and clock signals arrive at the receiver at the same time. If the sender core operates at a higher frequency than the receiver, it must not send new data while the receiver is not ready to read new data. A rate controller is therefore required on the sender side to enforce this condition. The implementation of a rate-controller is straightforward [4]. Notice that since the rate-controller does not require any input from the receiver side, there is no possibility of a metastable event in it.

The following statements about DCM correctness and performance are given without proof due to space limitations.

LEMMA 1. *If the setup and hold constraints of the interface latches are satisfied, then the probability of synchronization failure (and metastability) is zero.*

THEOREM 2. *DCM can be implemented to satisfy Lemma 1.*

THEOREM 3. *DCM achieves maximum throughput.*

3. SIMULATION RESULTS

We implemented a full-custom design of DCM in a $0.13\mu m$ CMOS process. Using representative latches, Hspice simulations of two communicating cores were performed: One representing a CPU running 500MHz and another corresponding to a peripheral operating at 333.3MHz.

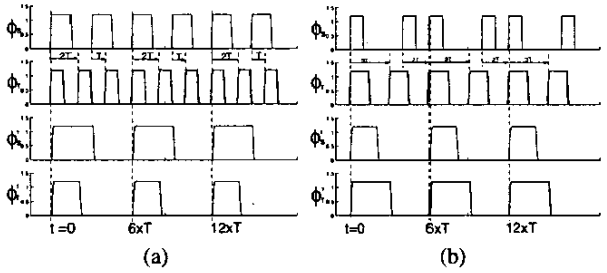

Figure 5: Hspice simulation waveforms: (a) $n < m$,(b) $n > m$.

The first experiment used the topology shown in Figure 2. Waveforms from Hspice simulation sweeps are shown in Figure 5(a). Regardless of the initial phase shift in the receiver clock, the ϕ_r waveforms coincide well, as the PCM inserts the appropriate delay in the clock tree of the receiver core to ensure edge-alignment. The ϕ_r waveform is aligned with ϕ_s every three cycles. The PCA module is thus verified over the entire range of receiver clock phase shifts (not shown).

The second experiment simulated data transfers from the CPU to the bus. Figure 5(b) shows the waveforms obtained from Hspice simulation sweeps. The rate controller clock-gates the sender latch in the event that previous data has not been latched by the receiver. Periodic clock alignment ensures that the minimum time difference to cause a setup time violation is at least T.

4. REFERENCES

[1] D. M. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford, October 1984.

[2] W. Muttersbach, T. Villiger, and W. Fichtner, "Practical design of globally-asynchronous locally-synchronous systems," in *ASYNC'00*, pp. 52–59, April 2000.

[3] K. Yun and R. Donohue, "Pausible clocking: a first step toward heterogeneous systems," in *ICCD'96*, pp. 118–123, October 1996.

[4] A. Chakraborty and M. Greenstreet, "Efficient self-timed interfaces for crossing clock domains," in *ASYNC'03*, pp. 78–88, May 2003.

[5] R. Ginosar, "Fourteen ways to fool your synchronizer," in *ASYNC'03*, pp. 89–96, May 2003.