On-chip synchronous communication between clock domains with quotient frequencies

V. Sathe, M.C. Papaefthymiou, S.V. Kosonocky and S. Kim

Data transfer between arbitrary asynchronous clock domains is complicated, and practical solutions can be subject to synchronisation failure. But many practical systems use quotient frequencies and the direct conversion method (DCM) exploits this to achieve reliable data transfer without any handshaking. DCM minimises throughput, even between distant cores, is provably correct and has been assessed through transistor-level simulations.

Introduction: The continuing trend in system-on-a-chip (SoC) applications is towards ever-increasing functionality, performance and integration, leading to systems which dissipate hundreds of Watts [1]. A popular approach to power saving is to split a SoC design into many clock domains, so that each domain only runs when it is required, and at as low a clock rate as possible. However, the use of multiple frequencies requires robust and efficient inter-core communications that are immune to synchronisation failure when transferring data across clock domains.

Previous approaches to the crossing of clock domains have considered arbitrary clock frequencies [2–5], but this generality results in reduced throughput, especially over long-latency channels. In many SoCs, however, clock domains are generated from a single master clock using dividers, and this can be exploited to enable efficient and provably correct inter-core communication between clock domains running at different frequencies.

We introduce an on-chip communication technology, which we call the direct conversion method (DCM), that is specifically aimed at SoCs with multiple clock domains derived by dividing a master clock. DCM synchronises the communicating cores directly, without using handshaking protocols, and this provides several advantages over existing interface techniques: provably zero probability of synchronisation failure; minimal communication latency in inter-core communication across long distances; maximum communication throughput, regardless of the physical separation of the two cores; and straightforward implementation based on widely used circuit techniques. We have proved the correct operation of DCM analytically, and its efficiency and performance have been assessed through HSpice simulations of two cores implemented in a 0.13 µm CMOS process and running at 500 and 333.3 MHz.

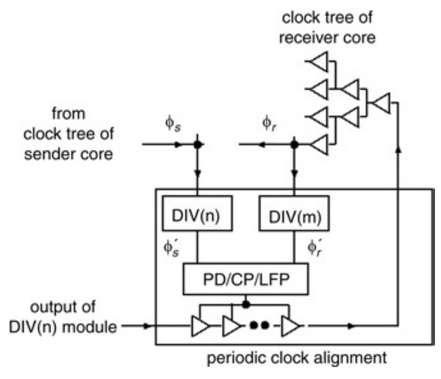

Fig. 1 *Dynamic tuning for periodic clock alignment*

On-chip communication between clock domains: DCM is applicable to SoCs with clock domains derived from a master clock frequency through integer division. Since the ratio between the resulting clock periods is rational, the latching edges of two clock domains that align at some time are guaranteed to realign periodically thereafter. Furthermore, it can be shown that the time that elapses between a clock edge in the sending domain and a non-coinciding clock edge in the receiver is at least one full-time period of the master clock. This ensures reliable synchronisation, without any handshake circuits between the clock domains, provided that simple timing constraints are met.

Periodic alignment of two coincident clock edges can be enforced by dynamic tuning of a delay line. The control circuitry used to perform this tuning in DCM is called the periodic clock alignment (PCA) module, which divides the periods from both clock domains to a least common multiple-period. Then the PCA tunes the delay line to correspond to the phase difference between these derived frequencies.

Fig. 1 shows the PCA module. The sender clock Φ_s and the receiver clock Φ_r have 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 values of m and n 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 m and n, respectively, so that the period of both derived clocks, Φ_s' and Φ_r', is $m \cdot n \cdot T$. At periodic intervals of $m \cdot n \cdot T$ the PCA module uses the signals Φ_s' and Φ_r' to tune the delay applied to Φ_r to align the clock edges of Φ_s and Φ_r. The remaining modules, the phase detector (PD), the charge pump (CP), and the voltage-controlled delay line (VCDL), are all standard components of a conventional DLL.

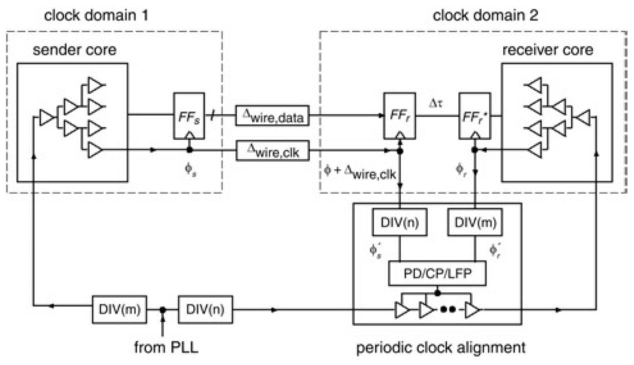

Fig. 2 *DCM between distant cores*

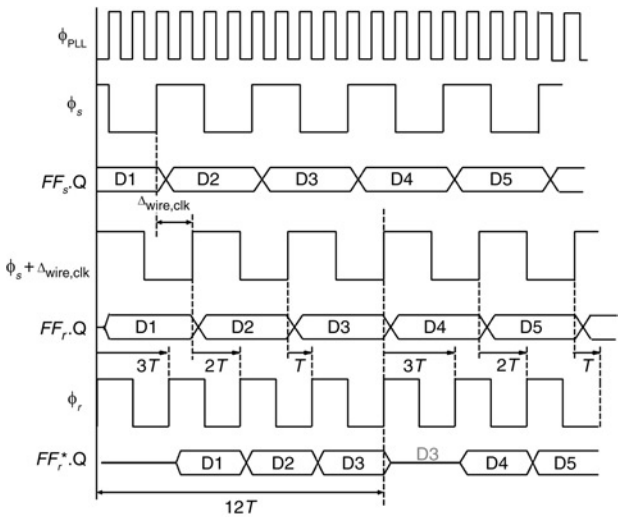

Fig. 3 *Data flow across clock domains ($m = 4$, $n = 3$)*

Fig. 2 illustrates the operation of DCM when two cores are at an appreciable distance from each other. The latch FF_r is physically located in the receiver clock domain and is clocked by the sender's clock modified by an allowance for the wire delay: $\Phi_s + \Delta_{\text{wire,clk}}$. To avoid a setup time violation during data transfer, the following inequality must always be satisfied for every instance k of the receiver clock edge:

$$[(k \cdot n) \bmod m] \cdot T > T_{clk-q,FF_r} + T_{setup,FF_r^*} + \Delta\tau \qquad (1)$$

where T_{clk-q,FF_r} is a nominal clock to output delay of FF_r, T_{setup,FF_r^*} is the setup time of FF_r^*, and $\Delta\tau$ is the combined uncertainty of the clock edges of Φ_s and Φ_r. In particular, this inequality must hold for the minimum positive value of $[(k \cdot n) \bmod m] \cdot T$. By considering $(k \cdot n)$ mod m as the result of a linear combination of integers, it is straightforward to show that the minimum nonzero difference between the rising edges of the two clocks Φ_s and Φ_r is $\eta \cdot T$, where η is the greatest common divisor of m and n. Therefore, considering the worst-case scenario when $\eta = 1$, inequality (1) reduces to

$$T > T_{clk-q,FF_r} + T_{setup,FF_r^*} + \Delta\tau \qquad (2)$$

FF_r must be adjacent to FF_r^* in order to minimise $\Delta\tau$. The timing diagram of the transmitted data for $m = 4$ and $n = 3$ is shown in Fig. 3.

The time difference between a write edge of FF_r and the next read edge of FF_r^* is at least T. Observe that D3 is latched twice by FF_r^*. A rate controller in the sender and receiver is necessary to enforce this condition, but its implementation is straightforward [4]. Since it does not require any input from the receiver side, there is no possibility of a metastable event in the rate controller.

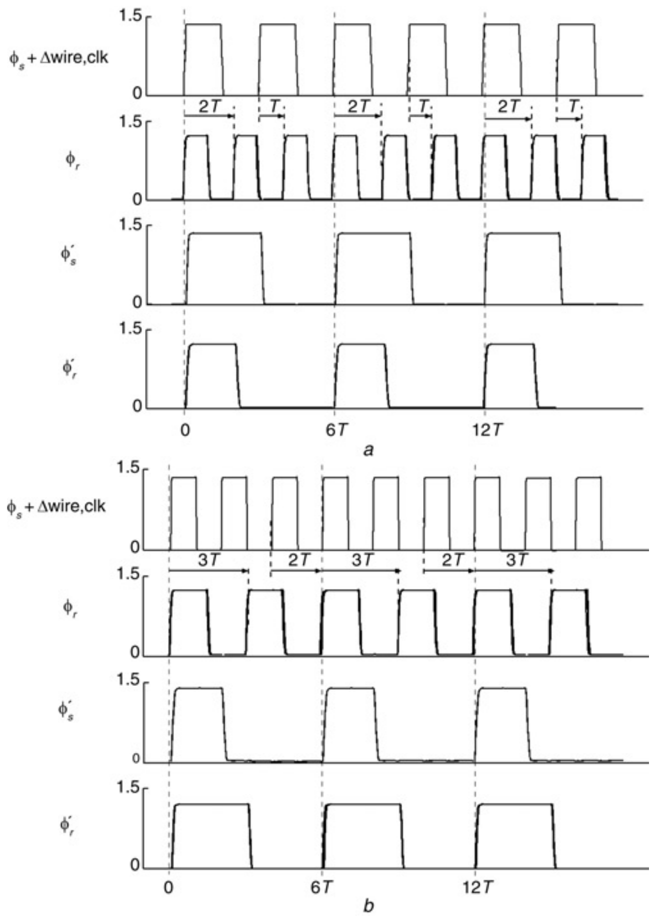

Fig. 4 *Waveforms from transistor-level simulation*

a $n<m$
b $n>m$

Simulation results: We have implemented a fully customised DCM design in a CMOS process. Using representative latches, HSpice simulations of two communicating cores were created: one models a CPU running at 500 MHz and the other corresponds to a peripheral operating at 333.3 MHz. The first simulation used the topology shown in Fig. 2. Waveforms from HSpice sweeps are shown in Fig. 4*a*. Regardless of the initial phase shift in the receiver clock, the waveforms coincide closely because 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 $\Phi_s + \Delta_{\text{wire,clk}}$ every three cycles, thus verifying the PCA module over the entire range of receiver clock phase shifts. The second experiment simulated data transfers from the CPU to the bus. Fig. 4*b* shows the waveforms obtained from HSpice sweeps. The rate controller clock-gates the sender's latch if previous data has not been latched by the receiver.

Conclusions: The direct connection method (DCM) is a novel technique for crossing clock domains which involves minimal latency and allows full throughput. The probability of failure is reduced to zero because the method exploits the way in which multiple clock domains are usually generated in SoC designs. We have demonstrated the correct operation of DCM by simulating a custom implementation in a 0.13 µm CMOS process. DCM offers the designer additional flexibility in the choice of operating frequencies since it will work for any clock domains that are generated from the PLL clock by rational division, and not just for integer divisors. It is straightforward to extend the analysis to this case. Because the delay lines are tuned dynamically, DCM is inherently able to cope with changes of master clock frequency. If, in addition, the elements of the interface circuitry have timing parameters that are relatively insensitive to voltage scaling, the use of DCM would allow DVS to be applied to a SoC with many domains without requiring explicit concern about the reliability of the interfaces between them.

8 January 2007
Electronics Letters online no: 20070057
doi: 10.1049/el:20070057

V. Sathe and M.C. Papaefthymiou (*Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA*)

S.V. Kosonocky (*IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA*)

S. Kim (*Electrical Engineering, Seoul National University, Seoul 151-744, Korea*)

E-mail: suhwan@snu.ac.kr

References

1 Nowka, K.J., Carpenter, G.D., and Brock, B.C.: 'The design and application of the PowerPC 405LP energy-efficient system-on-a-chip', *IBM J. R&D*, 2003, **47**, (5–6), pp. 631–639
2 Muttersbach, W., Villiger, T., and Fichtner, W.: 'Practical design of globally-asynchronous locally-synchronous systems'. ASYNC, April 2000, pp. 52–59
3 Yun, K., and Donohue, R.: 'Pausible clocking: a first step toward heterogeneous systems'. ICCD, October 1996, pp. 118–123
4 Chakraborty, A., and Greenstreet, M.: 'Efficient self-timed interfaces for crossing clock domains'. ASYNC, May 2003, pp. 78–88
5 Ginosar, R.: 'Fourteen ways to fool your synchronizer'. ASYNC, May 2003, pp. 89–96