Synchronisation methods for multi-synchronous systems

George Taylor, Robert Mullins
Computer Laboratory, University of Cambridge
{*gst22,rdm34*}*@cl.cam.ac.uk*

Overview

This talk provides a brief overview of synchronisation schemes which might be deployed in a multi-synchronous system. The basic premise is that synchronisation incurs a cost. Synchronisation schemes and circuits, all obtained from existing literature, are used to highlight some of the tradeoffs involved in allocating that cost. Taking a multi-synchronous perspective may be the key to the management of integration and energy dissipation for large system on a chip designs. This talk will raise questions about the future role of a multi-synchronous methodology. Some background concepts are summarised below.

Communication, synchronisation and timing disciplines

All communication involves some kind of synchronisation. Synchronisation involves a cost. That cost can be paid in a number of ways, for example: latency, MTBF (mean time between failure), energy. Perhaps more interestingly different parts of that cost can be paid at different times: once at initialisation, periodically, at each communication burst, or at each communication.

Related to the type of synchronisation is *timing discipline*. Timing disciplines arise from the need to sequence computation in an implementation where delays can not be completely predetermined or vary in a non-trivial manner. Here are some examples of terminology used to describe timing within a system: *Synchronous*—a single clock, ideally with no skew or jitter. *Multi-synchronous*—multiple clock domains with various relationships. *Handshaking*—for example bundled-data, speed-independent, delay-insensitive (most UK forum readers refer to these as asynchronous).

Synchronisation of asynchronous events

Synchronising two or more asynchronous events can be attempted in two ways:

bounded-time Assume metastability resolves in bounded time and pay with mean time between failure. In this case failure might involve parts of the system becoming metastable resulting in incorrect state. Upon failure recovery of the system is non-trivial. Latency and MTBF can be traded for one another. Interfacing asynchronous inputs to a synchronous system with a (non-stretchable) periodic clock is a bounded-time synchronisation.

unbounded-time Wait for metastability to resolve and pay with a theoretically unbounded execution time, and more practically, less predictable performance. Although the circuit will not enter an incorrect state, if a hard performance guarantee is required there is still a mean time between failure. For example an unbounded-time synchroniser could not transmit to a periodic clocked domain. The use of an arbiter in handshake circuits, or a stretchable clock, is an unbounded-time synchronisation.

Multi-synchronous systems

A multi-synchronous system is where there exists either ideal, or more practically, a suitable approximation to, synchronous islands in an ocean of asynchrony. The table below lists possible relationships between clock domains *at the point of communication* characterised by the frequency and phase. Many of these terms originate in the telecommunications world.

name	**frequency**	**phase**
synchronous	same	same
mesochronous	same	constant
plesiochronous	small difference	slowly varying
derived	fixed simple ratio	constant
related	fixed simple ratio	slowly varying
heterochronous	different	constant/varying
asynchronous	non-periodic	unknown

It is worth noting that although the clocks in different clock domains might follow a particular relationship it is really the synchronisity of the communication between them that is captured in the above table. The communication mechanism might hide any preexisting relationship between the clock domains. As an extreme example two synchronous domains might communicate via an asynchronous chip-wide packet switching network with varying communication latency – as far as synchronisation for communication is concerned the two domains are asynchronous. The term globally-asynchronous locally-synchronous (GALS) is often used to describe a VLSI system of multiple clock domains where communication is, for whatever reason, asynchronous.

In general the problem with synchronising asynchronous inputs is that the time of input can not be predicted, and thus the time for metastability resolution is paid in communication latency. However, in the synchronous to heterochronous cases listed above it is possible to take advantage of the periodicity of the clocks in the transmitter and receiver. It becomes possible to predict the receiver clock for each cycle of the transmitter clock, and thus the transmitter can avoid transmitting during the *keep-out region* of the receiver. Because this prediction can be made in advance, the time for metastability resolution is removed from the communication latency, but possibly at the, hopefully smaller, expense of delaying the data until communication is safe. Discussion and examples of the use of prediction include [1–3].

All of the schemes summarised below aim to transfer data from a transmitter to receiver without discarding items or creating duplicates. In a real-time control environment, for example where quantised digital representations of continuous analogue values are to be transmitted, it is possible to avoid blocking due to flow control and have a hard performance guarantee at the expense of discarding data items or obtaining duplicates [4].

Synchronous Communication is synchronised by definition. The cost of synchronisation is paid in clock distribution and related safety margins. For a large high frequency system this cost may become non-trivial. Guaranteeing performance is straightforward. The clocks need not be periodic but must have near-constant phase difference, for example skew in distribution from a single source. Metastability is avoided.

Mesochronous In a mesochronous system synchronisation needs only to be performed once at initialisation. Guaranteeing performance is straightforward. Metastability can occur only at initialisation permitting ample time for resolution with a low MTBF. Such a scheme is comparable to adaptive de-skewing in clock distribution [5]. In practice, due to environmental drift, it may be desirable to re-synchronise from time to time.

Plesiochronous A plesiochronous system can be viewed as a mesochronous system but periodic re-synchronisation is required. To avoid duplicating or discarding data items, insertion of null

spacers or flow control must be used. Performance guarantee is possible provided the difference in clock frequencies is known. It must be stressed that the difference in frequency is small.

Derived and related These are related to mesochronous and plesiochronous except that the clock frequencies are some ratio of one another. In the derived case the clock cycles on which communication is safe can be calculated algorithmically and thus *metastability is avoided* [6].

Heterochronous Like the plesiochronous case a predictive synchroniser can be used, here the transmitter must predict the receiver clock sufficiently far in advance. There is a tradeoff between time for metastability to resolve and how far into the future it is possible to predict. If the clock frequencies are known it is possible to predict performance.

Asynchronous It is not possible to remove the latency cost of metastability resolution, however, bandwidth may be kept high by use of pipelining [7] or multiplexers and multiple synchronisers. Latency is traded for MTBF and is unavoidable. Alternatively stretchable clocks can be used, for example [8, 9]. This case permits the mixing of synchronous and self-timed logic blocks.

The selection of a synchronisation scheme to use in a multi-synchronous system is likely to depend on many specific factors. For example, latency may be important in a general purpose processor, but not in a streaming media processor. Or the choice of communication mechanism, for example point to point or packet switching network. The future of a multi-synchronous methodology relies upon the clear identification of applications and choice of communication mechanisms.

References

[1] W. J. Dally and J. W. Poulton, *Digital Systems Engineering*. Cambridge University Press, 1998.

[2] D. G. Messerschmitt, "Synchronization in digital system design," *IEEE Journal on Selected Areas in Communication*, vol. 8, no. 8, pp. 1404–1419, 1990.

[3] F. Mu and C. Svensson, "Self-tested self-synchronization circuit for mesochronous clocking," *IEEE Transactions on Circuits and Systems, II: Analog and Digital Signal Processing*, vol. 48, pp. 129–140, Feb. 2001.

[4] H. Simpson, "Four-slot fully asynchronous communication mechanism," *IEE Proceedings, Part E, Computers and Digital Techniques*, vol. 37, pp. 17–30, 1990.

[5] M. Saint-Laurent and M. Swaminathan, "A multi-PLL clock distribution architecture for gigascale integration," in *Proc. IEEE Computer Society Workshop on VLSI*, pp. 30–35, Apr. 2001.

[6] L. F. G. Sarmenta, "Rational clocking," in *Proc. International Conf. Computer Design (ICCD)*, IEEE Computer Society Press, 1995.

[7] J. N. Seizovic, "Pipeline synchronization," in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 87–96, Nov. 1994.

[8] D. M. Chapiro, *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, Oct. 1984.

[9] F. U. Rosenberger, C. E. Molnar, T. J. Chaney, and T. P. Fang, "Q-modules - internally clocked delay-insensitive modules," *IEEE Transactions on Computers*, vol. C-37, pp. 1005–1018, Sept. 1988.

This work is funded by EPSRC grant GR/R52299/01.