Quasi-Delay Insensitive Bus for Fully Asynchronous Systems

Pedro A. Molina
pamolina@ee.ic.ac.uk

P. Y. K. Cheung
p.cheung@ic.ac.uk

Department of Electrical and Electronics Engineering
Imperial College of Science, Technology and Medicine
Exhibition Road, London SW7 2BT
United Kingdom

Extended Summary

Introduction

During the past 10 years a revive interest in asynchronous circuits has emerged as a mean to overcome some of the design difficulties presented by the sub-micron and sub-nanosecond VLSI technology available today. As the transistor feature size decreases, VLSI designers are now incorporating millions of transistors within a single chip. This increase in density has also been accompanied by a significant reduction of the switching speed at the gate level. All these technological improvements do not come free and designers are now face with some very difficult design issues. The interconnection delay is becoming a major problem, demanding extensive post layout timing simulation and very complex schemes to distribute the global clock signal. The limitations of the synchronous paradigm have been widely documented (see for example [Sei84].) The most commonly referred disadvantages of synchronous circuits are: 1. The maximum frequency of operation is limited by the worst case timing on the critical path and by the worst case clock skew within the system; 2. There is a waste of power by the constant toggle of the clock line in portions of the circuit which are not required at a given time, and 3. Synchronous circuits exhibit little composability.

In contrast to these problems, asynchronous circuits could potentially exhibit an average or best case delay, low power consumption, ease of global timing issues, better technology migration potential and automatic adaptation to physical properties [Hauc95]. Several new asynchronous techniques have been proposed. They, have been successfully applied to the design of some fully asynchronous integrated circuits (see for example [Furb95][Kess95][Will91]). Although much research is still needed, the results are very encouraging, with some designs having made significant improvements in speed and power consumption.

In this paper, however, we will focus on an additional metric, one which is constantly mentioned as a potential advantage of asynchronous systems, but which is usually neglected on the performance evaluation of a system. This fourth metric is commonly referred as the "modularity" or "composability" principle [vBer93]. Analogous to software engineering, in which object oriented programming allows easy "interconnection" and "reusability" of software modules, hardware designers would like to be able to interconnect different, previously designed, hardware modules without incurring long design cycles. Within the synchronous paradigm the composability principle is difficult to meet, and many post-layout timing verification and re-routing iterations are often required.

The asynchronous paradigm should, in principle, provide a great amount of composability and, therefore, reduce considerably the design time of a VLSI system. However, if conventional delay insensitive techniques are blindly applied to a large system, the final circuit will normally consume a prohibitively large amount of silicon area. This penalty arises primarily from the difficulty of providing dedicated data paths for all the interconnected modules. A popular technique to alleviate this problem is to use one or more common buses. By doing this, a single set of wires is used to interconnect several blocks within the architecture. This approach offers the benefit of increasing the complexity of the system linearly with each new module added to the system. Synchronous architectures will normally use tri-state logic to form the bus. However, the use of tri-sate logic is not straightforward when implementing quasi delay insensitive circuits and designers have avoided them altogether [Nany94] or used very wide distributed OR gates [Mart89]. Specifically, two major obstacles are apparent when trying to use tri-sate logic within an asynchronous circuit: violation of the isochronic fork assumption and the non-deterministic behaviour of "floating" logic (the former problem is identified in [Nany94], while the latter is reported in [Cho92]).

Design of a Dual-Rail Quasi Delay-Insensitive Bus

In this section we present the design of a system bus for an asynchronous architecture which overcomes the problems described above. Because there is no global clock, each transaction on the bus should be self-timed. That is, a handshake protocol should be established between the sender and the receiver. A bus typically has many branches, with each of them varying considerably in length and capacitance. Therefore, the propagation delay from any two pairs of nodes in the bus also varies considerably. To compensate for these differences, a common solution is to use a delay insensitive code on the bus [Verh88]. Of all the possible codes the most popular is the so called Dual-Rail Return-to-Zero. In this code, two wires are used for every bit in the bus. The two wires encode the value of the corresponding bit, where the value 01 denotes a binary zero and 10 denotes a binary one. The code 00 is used as a spacer and should always be present between two bit values. By using this code, a destination port will always know when there is a valid word on the bus no matter how long it takes the signal to propagate from the

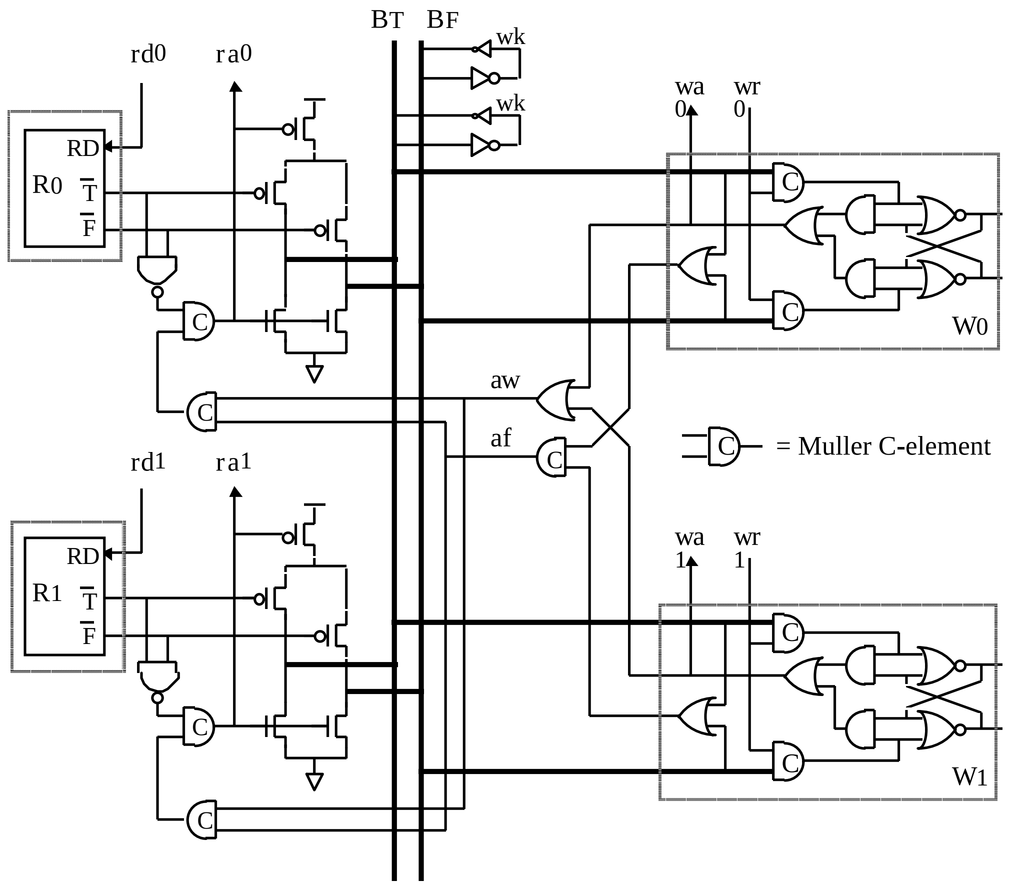

Figure 1. Example of a one bit quasi-delay insensitive bus. The input and output registers are explicitly shown for clarity but are not part of the bus circuitry (see text)

sender. Also, “nearby” nodes can latch the bus value as soon as it arrives and, therefore, do not have to pay the worst case propagation time for the entire bus (as is required in a synchronous circuit).

The biggest problem for the implementation of a dual rail bus has to do with the relinquishing of the bus. Once a receiver acknowledges the transaction, the corresponding sender, should stop driving the bus so that the control unit can select a new pair of nodes for the next transaction. When the control unit turns over the bus from one sender to another, three main issues should be considered. First, enough control logic should be incorporated to avoid two nodes from simultaneously driving the bus, and therefore producing a charge sharing problem (this problem was identified in [Cho92]). Second, when the bus is not driven by any node, provision must be taken to guarantee that the charge in the bus is retained, otherwise, the delay insensitivity of the code would be lost. Finally, before a new cycle starts, it should be guaranteed that all the branches in the bus have gone back to zero as to avoid the subsequent selected destination node from erroneously latching the data from the previous cycle

We have designed all the required modules to build a quasi-delay insensitive dual rail bus [Mol96]. To illustrate the principle of operation we offer figure 1 as an example of a one bit bus. In the example two read registers (R_0 and R_1) and two write registers (W_0 and W_1) are shown.

Signals rd_i and ra_i are the read and read acknowledge signals respectively for port R_i, while wr_j and wa_j, are the write and write acknowledge handshake signals for the write port W_j. All these signals are fed into the control unit (not shown in the figure).

A typical bus cycle between source node R_i and destination node W_j is as follows (see also figure 2): The control unit activates signals rd_i and wr_j concurrently. When register R_i has valid data, it will pull up the required bus wire. This value will propagate along the bus and, because of the delay insensitive code used, register W_j will detect the arrival of valid information. After latching the new value, register W_j will acknowledge the cycle through signal aw_j. However, before starting the return to zero phase, register R_i must wait for the whole bus to be stable. This is done with a Joint operation [Suth89] (provided by the C-element) of all the fork acknowledges (a_f) coming from all the destination ports. After this, port R_i will activate the pull-down and will start the return to zero phase. Again, port R_i will wait for all destinations port to signal that the bus has return to zero. At this point R_i stops driving the pull down on the bus and the control unit is signalled to indicate that it is now safe to initiate a new transaction.

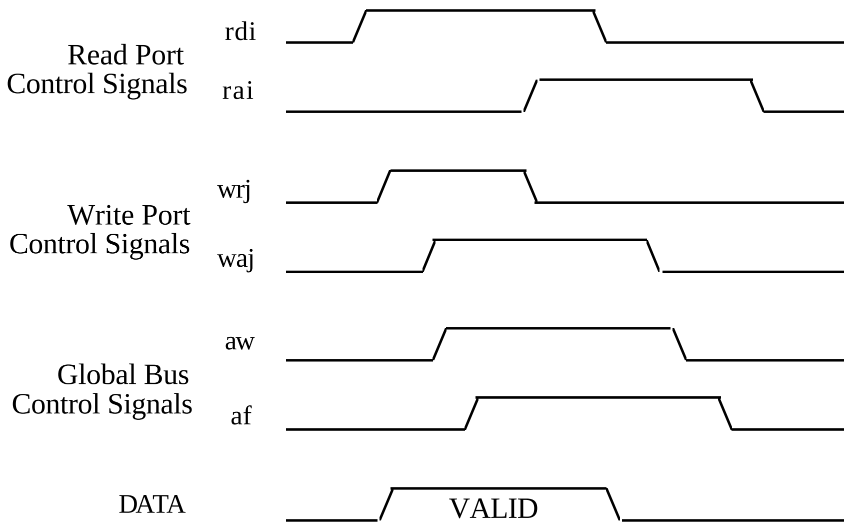
Read Port Control Signals
rdi
rai
Write Port Control Signals
wrj
waj
Global Bus Control Signals
aw
af
DATA
VALID

Figure 2. Timing diagram for a transaction between output port R_i and input port W_j (see text)

Notice how in between two transactions, the bus is not driven. To prevent the bus from loosing its charge, and therefore violating the delay insensitive code, weak inverters are provided on every bus line to conserve the charge [Cho92].

Extension to an N-bit bus is achieve by the replication of the components shown in figure 1. It is common practice to use a single acknowledge signal for an N-bit bus (instead of N signals). In general, completion trees should be provided to merge the completion of all bits within a single signal. However, in many cases a single bit in the bus can be used to signal the completion of the whole word [Meng89]. Also, but beyond the scope of this paper, the circuitry of the register can be reduced by applying an extension to the isochronic fork assumption as proposed in [vBer95].

Conclusions

Traditionally designers of asynchronous circuits have avoided the use of tri-state buses due to the undesirable effects of the heavily loaded lines and the use of “floating” logic. In this paper, however, we have shown that it is possible, within some reasonable timing assumptions, to design a quasi-delay insensitive bus structure.

The solution presented in this paper offers the benefit of exploiting the composability principle of asynchronous circuits. Modules can easily be added and removed from a system with very little additional design effort. Also, the complexity of the circuit grows linearly with each new module added to the architecture.

References

[Cho92] K. Cho, K. Okura and K. Asada, “Design of a 32-bit Fully Asynchronous Microprocessor (FAM)”, In Proceedings of the 35th Midwest Symposium on Circuits and Systems, IEEE, New York 1992, pp. 1500-1503.

[Furb95] S. Furber, “Computing without Clocks: Micropipelining the ARM Processor”. In G. Birtwistle and A. Davis, editors, *Asynchronous Digital Circuits Design*. Springer, 1995

[Hauc95] S. Hauck, “Asynchronous Design Methodologies: An Overview”, Proceedings of the IEEE, 83(1):69-93, January 1995.

[Kess95] J. Kessels, “VLSI Programming of a Low-Power Asynchronous Reed-Solomon Decoder for the DCC Player”, In Proceedings of the Second Working Conference on Asynchronous Design Methodologies, South Bank University, London, May 1995

[Mart89] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus, “The design of an asynchronous microprocessor,” in *Advanced Res. VLSI: Proc. Decennial Caltech Conf. VLSI*. Cambridge, MA: M.I.T Press, Mar. 1989

[Meng89] T. H. Y. Meng, R. W. Brodersen, D. G. Messerschmitt, “Automatic Synthesis of Asynchronous Circuits from High-Level Specifications”, IEEE Transactions on CAD, Vol. 8, No. 11, Nov. 1989, pp. 1185-1205.

[Mol96] P. Molina, P. Cheung and T. King-Smith, “Design and Analysis of a Self-Timed Bus”, paper to be submitted to the Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Aizu University, Japan, March 1996.

[Nany94] T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako, A. Takamura, “TITAC: Design of a Quasi-Delay-Insensitive Microprocessor”, IEEE Design & Test of Computers, Aug. 1994, pp. 50-63.

[Sei84] Charles L. Seitz. System Timing. In C.A. Mead and L.A. Conway, editors, *Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1980

[Suth89] I.E. Sutherland, “Micropipelines”, *Communications of the ACM,* Volume 32, Number 6, June 1989, pp. 720-738

[vBer93] K. van Berkel, “Handshake Circuits, An asynchronous architecture for VLSI programming”, Cambridge International Series on Parallel Computing 5, Cambridge University Press. 1993.

[vBer95] K. van Berkel, F. Huberts, A. Peeters, “Stretching Quasi Delay Insensitivity by Means of Extended Isochronic Forks”, In Proceedings of the Second Working Conference on Asynchronous Design Methodologies, South Bank University, London, May 1995

[Verh88] T. Verhoeff, “Delay-insensitive codes - an overview,” Distributed Computing 1988, 3:1-8

[Wil91] T. Williams “A Zero-Overhead Self-Timed 160-ns 54-b CMOS Divider” IEEE J. of Solid State Circuits V 26 N 11, Nov. 1991 pp1651-1661