Testing the data freshness properties of asynchronous communication mechanisms

F. Xia[*] and A. Yakovlev
Department of Computing Science, University of Newcastle upon Tyne
I.G. Clark
Department of Electronic Engineering, King's College London

Indexing terms: asynchronous data communications, concurrent systems, data freshness, data sequencing.

1 Abstract

Data freshness properties of asynchronous communication mechanisms (ACMs) are related to the relative timing of reader and writer statements. When a hardware implementation of an ACM is tested at the highest operating speed, such relative timing is difficult to measure if the observing hardware is not constructed of faster technology than the ACM. Here an indirect method is proposed whereby the data freshness properties of any ACM hardware may be inferred from data sequencing information.

2 Introduction

Data communication between processes running in different processors has been extensively studied, especially in distributed and real-time systems. The minimal form of this problem concerns the passing of data between two distributed single-thread processes. One provides the data, which is used by the other. This is schematically shown in Figure 1.

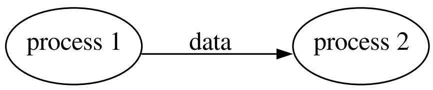

Figure 1 Passing data between two processes.

In distributed systems, processors may not always share a common clock. When the two processes in Figure 1 are not synchronised, some kind of intermediate data repository, usually in the form of shared memory, is often needed between them to facilitate the data passage. This is schematically shown in Figure 2.

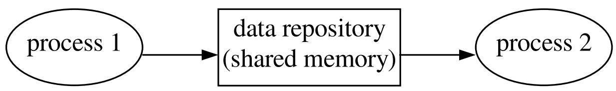

Figure 2 Passing data via shared memory.

An asynchronous data communication mechanism (ACM) is a scheme which manages the transfer of data between two processes not necessarily synchronised for the purpose of data transfer. It is assumed that the data being passed consists of a stream of individual items of a given type. It is also assumed that the processes in question are single thread cycles, one providing an item of data during each cycle, the other making use of an item of data during each cycle. The provider of data is known as the "writer" of the ACM and the user of data is known as the "reader" of the ACM.

Such techniques as the multiple "slot" (or "track") mechanisms described in [1], [2] and [3] realise regular and atomic registers in the data path between asynchronous concurrent processes by employing "safe" bit registers to convey the values of bit-size control variables. They avoid conflicts on data memory without resorting to explicit critical sections. The general scheme of these data communication mechanisms is shown in Figure 3.

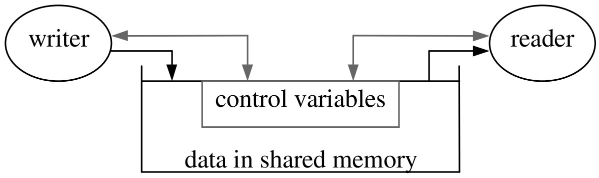

Figure 3 ACM using shared memory and control variables.

One of the most important properties of such ACMs is data freshness, which indicates how up to date any item of data obtained by the reader is, relative to the latest item of data provided by the writer. Since

[*] Contact: fei.xia@ncl.ac.uk

both reader and writer processes tend to include multiple statements, the definition of data freshness is rather complicated. However, data freshness has proved to be verifiable both theoretically [4] and with digital simulations [5].

On the other hand, data freshness is almost impossible to directly verify from hardware testing if the hardware implementation of an ACM uses fastest available VLSI technology.

Here, the relationship of data freshness and data sequencing properties of ACMs is closely studied and a method derived whereby data freshness properties can be inferred from data sequencing information, which is easy to observe in hardware testing regardless of the hardware technology involved.

3 Background studies

A particular ACM from the literature has been chosen as a basis to illustrate the problem and solution. This is briefly described. The two pertinent ACM properties are defined.

3.1 The 4-slot ACM algorithm

Of the multiple slot or track ACMs that provide fully asynchronous operations, by far the simplest and easiest to implement in hardware is the Simpson's 4-slot mechanism found in [1].

Writer	Reader
wr: $d[n, \overline{s[n]}] := input$	$r0$: $r := l$
$w0$: $s[n] := \overline{s[n]}$	$r1$: $v := s$
$w1$: $l := n \parallel n := \bar{r}$	rd: $output := d[r, v[r]]$

Figure 4 Simpson's 4-slot mechanism.

Figure 4 shows the algorithm of the 4-slot mechanism. Here the writer and reader processes are single thread loops with three statements each. The mechanism maintains the storage of four data slots $d[0, 0]$ to $d[1, 1]$ and the control variables n, l, r, $s[0..1]$, and $v[0..1]$, which are either single bits or vectors of two single bits. This is shown schematically in Figure 5. The statements wr and rd are the data accesses and the other statements are used by the writer and reader to choose slots and indicate such choices to the other side.

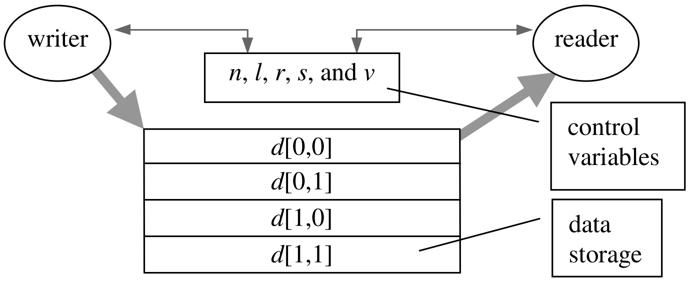

Figure 5 Schematic of the 4-slot mechanism.

3.2 Data freshness and data sequencing properties of ACMs

Considering the 4-slot ACM introduced above and similar algorithms, the data freshness and sequencing properties are defined below:

***Definition 1*. Latest and previous latest data items**

At any time, the latest item of data is found in the slot accessed by the last completed wr statement, and the previous latest item of data is found in the slot accessed by the last but one completed wr statement.

Data freshness issues are relevant only at the beginning of a reader access to a slot, i.e. at the beginning of the statement rd in Figure 4. Checking for data freshness obviously has to be done at this point.

***Definition 2*. Data freshness**

Data freshness is maintained if the reader always accesses an acceptable slot. Acceptable slots are:

1. the slot containing the item of data which was the latest at the beginning of the pre-sequence of the current reader cycle, $slot_L$;
2. the slot containing the item of data which is the previous latest at the beginning of the pre-sequence of the current reader cycle, $slot_P$, only if the pre-sequence of this reader cycle started while a writer post sequence was in progress;

3. the slot containing the latest item of data before the beginning of the current reader access (just before *rd*), $slot_{LL}$.

A reader cycle consists of one cycle of consecutive *r*0, *r*1 and *rd* statements. The statements *r*0 and *r*1 constitute the pre-sequence of the reader cycle to which they belong. The post sequence of a writer cycle consists of the statements *w*0 and *w*1.

In Definition 2, 1. is intuitive, as the reader access procedure uses the pre-sequence to determine where the latest item of available data is ($slot_L$), and a successful attempt at obtaining this item of data, even in the event of newer data becoming available during the reader pre-sequence ($slot_{LL} \neq slot_L$), cannot be regarded as a data freshness failure. In comparison, 3. is meant to cover the case in which the reader is after all able to obtain the location of $slot_{LL}$, even when $slot_{LL} \neq slot_L$, by design or by accident. It is recognised in 2. that if a writer post sequence and the pre-sequence of the current reader cycle overlap in time, the reader should not be expected to obtain the location of $slot_L$ because this information may not have been completely indicated by the writer post sequence yet. However the reader should be expected to at least obtain the location of $slot_P$ in this case, as that should have been indicated by the post sequence of the previous writer cycle.

***Definition 3*. Data sequencing**

Data sequencing is maintained if the reader never obtains data items in reverse order to that in which they were written into the ACM by the writer.

The 4-slot ACM in Figure 4 has been shown to maintain data freshness and data sequencing [4].

4 ACM hardware testing

During the development of a hardware implementation of the 4-slot ACM [5], the problem of testing for the important properties in hardware arose. Although such properties as data freshness and data coherence have been verified for the ACM both theoretically and via digital simulations, testing for such properties in hardware is necessary. Testing considerations are different for different properties. For instance, in [5], the individual reader and writer statements are enclosed within the ACM hardware itself and no ready interfacing to the outside is readily available for them. In addition, the sequencing of these statements are implemented with self-timed circuits, making it more difficult to correctly estimate their precise timing. Also, assuming that the ACM hardware is implemented with the fastest VLSI technology available and the testing is desired to be at its maximum performance, it would be almost impossible to construct hardware to observe the progress of the reader and writer statements precisely enough to verify any property depending on the timing relation between these statements. On the other hand, the ACM interfaces externally with the writer and reader once per cycle through standard start/done handshakes. This means that the relative timing of individual reader and writer cycles may be manipulated in testing. For instance, it is possible to make the reader and writer operate sequentially, rather than concurrently. It is also possible to adjust the relative lengths of the extra-ACM delays of the reader and writer [5].

4.1 Testing for data sequencing

Testing for data sequencing is straightforward. There is no requirement to monitor the precise timing of the reader and writer statements. The property can be derived directly from the data going into and coming out of the ACM. In order to force the ACM to work at maximum speed, the writer and reader hardware needs to be built on the same chip as the ACM and kept as simple as possible. A simple solution consists of a counter serving as the writer and a FIFO buffer serving as the reader. In testing, the counter can be made to produce a sequence of data items in ascending (or descending) value order and after the capacity of the counter and/or FIFO is exhausted, the system stopped and the data within the FIFO retrieved off line for analysis. Since the order of the data going into the ACM from the writing counter is known, that of the data retrieved from the FIFO can be compared with it to verify data sequencing.

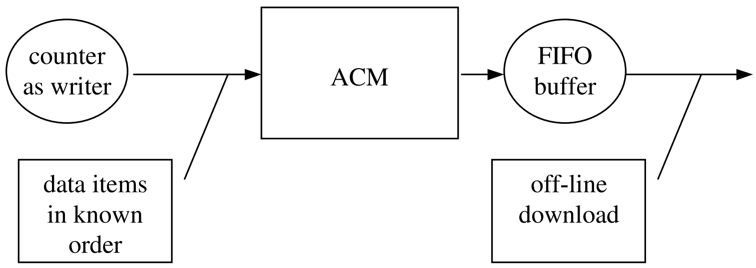

Figure 6 Testing for data sequencing.

Figure 6 shows the basic hardware needed to test for data sequencing.

4.2 Testing for data freshness

Since a direct testing of data freshness needs monitoring of the timing relationships between the reader pre-sequence and writer post sequence, much more needs to be done on-line in real-time than the case of testing for data sequencing.

Not only the relationship between the timing of the reader pre-sequence and writer post sequence must be recorded in every cycle of operation, the data values on both the reader and writer sides must also be recorded and time-stamped with regard to the pre- and post sequence statements. This requires a real-time clock to which all signals must be compared. Assuming that it is impossible to use faster circuits for this clock and the signal monitoring, this implies that an interfering observation must be used when the ACM is operating at maximum speed.

With this in mind and considering the ease with which data sequencing can be observed, it would be very advantageous if conclusions about data freshness may be drawn from observing data sequencing. From the definitions above, it is obvious that data freshness and data sequencing are not equivalent properties. However, within the scope of ACMs similar to the 4-slot, they are related to each other in subtle ways.

It is possible to maintain data freshness when data sequencing has been violated. For instance, a newer 4-slot ACM published in [6] maintains data freshness but not data sequencing [4]. However, this only happens under specific conditions.

Data sequencing is relevant only with regard to two consecutive cycles of the reader. It is violated if the reader obtains a data item which is older than the item it obtained in the previous cycle.

Assuming that the current reader and writer cycles are denoted with the letter c and the previous cycles are denoted with the letter p, the current writer post sequence then consists of the statements *w*0c and *w*1c.

The only situation where data freshness can be maintained while data sequencing is violated is when *r*0p and *r*1p overlapped with *w*0c and *w*1c and *rd*p was able to obtain the latest data item available at the time, i.e., that delivered to the ACM by *wr*c (conforming to 3. in Definition 2, thus maintaining data freshness). The reader cycle must then complete before the current writer post sequence is completed for *r*0c and *r*1c to also overlap with *w*0c and *w*1c and *rd*c to obtain the previous latest data item available, i.e., that delivered to the ACM by *wr*p (conforming to 2. in Definition 2, thus also maintaining data freshness). This is schematically shown in Figure 7.

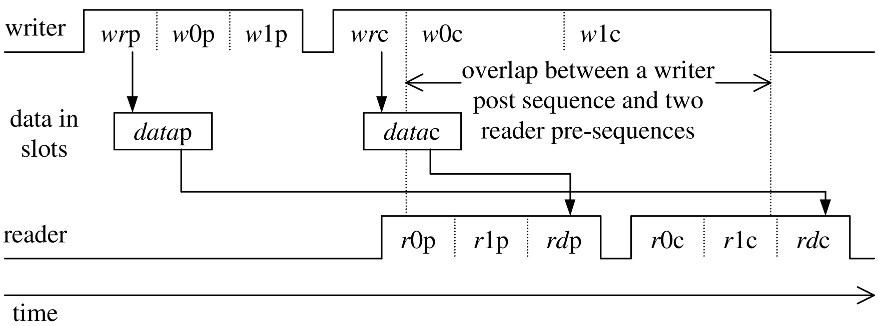

Figure 7 Data sequencing violated but data freshness maintained.

In other words, in order for data freshness to not imply data sequencing, the pre-sequences of two or more consecutive reader cycles must overlap with a single writer post sequence.

This can be avoided by running the writer at such a speed relative to that of the reader that a single writer post sequence is always shorter than a single *rd*. This can be implemented trivially by inserting appropriate delay elements into the reader and writer cycles.

Under this condition, any failure of data sequencing would indicate a failure of data freshness.

It is also possible for data freshness to be violated when data sequencing is maintained. The problem lies in latency. For instance, an ACM may cause the reader to always obtain the previous latest data item available. This does not cause data sequencing failures but does fail data freshness when there is no overlap of the writer post sequence and the reader pre-sequence. Latency is the only reason where data sequencing does not imply data freshness

Testing for the existence of latency in ACMs of types similar to the 4-slot is also a straightforward process. This can be done by running the writer counter for a number of times with known items of data produced and delivered to the ACM, without starting the reader action. After the writer is stopped, with the last item of data delivered to the ACM known, the reader is run a number of times, introducing this number of data items into the FIFO buffer. Since there is no overlap of any reader pre-sequence with a writer post sequence, data freshness means that all data items in the FIFO buffer should correspond to the last data item written by the writer. If this is not the case, the exact amount and pattern of latency can be derived by observing the data items found in the FIFO.

By combining the two tests above and through repeated testing, it is possible to qualitatively establish the absence of data freshness violations to a reasonable confidence level without manipulating and observing the exact timing relationship between the reader and writer statements.

5 Stronger definition of data freshness

For many practical applications, it can be said that when data sequencing is violated then stale, rather than fresh, data has been obtained by the reader, because a newer item of data was obtained during a previous cycle. It is easy to re-define data freshness so that this consideration is incorporated.

Definition 4. **Stronger data freshness**

Data freshness is maintained if the reader always accesses an acceptable slot. Acceptable slots are:

1. the slot containing the item of data which was the latest at the beginning of the pre-sequence of the current reader cycle, $slot_L$;
2. the slot containing the latest item of data before the beginning of the current reader access (just before *rd*), $slot_{LL}$;
3. the slot containing the item of data which is the previous latest at the beginning of the pre-sequence of the current reader cycle, $slot_P$, only if the pre-sequence of this reader cycle started while a writer post sequence was in progress and the previous reader cycle did not access its $slot_{LL}$.

This definition incorporates the notion of data sequencing and is much easier to test in hardware by observing data sequencing.

6 Conclusion and acknowledgement

A method has been proposed to overcome the difficulties encountered in testing the data freshness properties of ACMs in hardware. Appropriate hardware which support this kind of testing is being incorporated into the self-timed 4-slot design introduced in [5]. This work is part of the Comfort project at the University of Newcastle supported by the EPSRC grant GR/L93775.

References

1 Simpson, H.R., "Four-slot fully asynchronous communication mechanism", IEE Procs., Vol. 137, Pt. E, No. 1, pp.17-30, January 1990.

2 Tromp, J., "How to construct an atomic variable", Proc. 3rd Int. Workshop on Distributed Algorithms, Nice, LNCS, Springer Verlag, pp.292-302, 1989.

3 Kirousis, L.M., "Atomic multireader register", Proc. 2nd Int. Workshop on Distributed Computing, Amsterdam, LNCS-312, pp.278-296, Springer Verlag, 1987.

4 Xia, F. and Clark, I.G., "Complementing role models with Petri nets in studying asynchronous data communications", 19th International Conference on Application and Theory of Petri Nets, Hardware Design and Petri Nets Workshop, pp.66-85, Lisbon, Portugal, June 23, 1998.

5 Xia, F., Yakovlev, A., Shang, D., Bystrov, A, Koelmans A., and Kinniment D., "An Asynchronous communication mechanism using self-timed circuits", Technical Report, TR-686, Department of Computing Science, University of Newcastle upon Tyne, 1999.

6 Simpson, H.R., "New algorithms for asynchronous communication" and "Role model analysis of an asynchronous communication mechanism", IEE Proceedings on Computing and Digital Techniques, Vol. 144, No. 4, pp.227-240, July 1997.