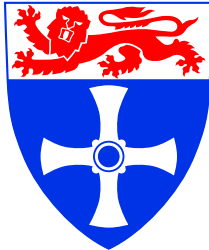


---

School of Electrical, Electronic & Computer Engineering

UNIVERSITY OF  
NEWCASTLE UPON TYNE



---

**Further discussions on the classification  
and high-level models of ACMs**

H. Simpson, E. Campbell  
F. Xia, I. Clark, A. Yakovlev, D. Shang

Technical Report Series  
NCL-EECE-MSD-TR-2004-102

---

2004

Contact:

Hugo.Simpson@mbda.co.uk

ERCampbell@iee.org

Fei.Xia@ncl.ac.uk

IGClark@iee.org

Alex.Yakovlev@ncl.ac.uk

Delong.Shang@ncl.ac.uk

NCL-EECE-MSD-TR-2004-102

Copyright © 2004 University of Newcastle upon Tyne

School of Electrical, Electronic & Computer Engineering,

Merz Court,

University of Newcastle upon Tyne,

Newcastle upon Tyne, NE1 7RU, UK

<http://async.org.uk/>

# **Further discussions on the classification and high-level models of ACMs**

H. Simpson, E. Campbell

F. Xia, I. Clark, A. Yakovlev, D. Shang

The main part of this report includes two short technical articles. These articles have been submitted to IEEE Micro for technical review and possible publication.

An article written by some of us (FX, AY, IC and DS) appeared in IEEE Micro towards the end of 2002 [1]. Parts of this article touch on the subject of classification of ACMs and their high-level modelling using Petri nets. The incompleteness and shortcomings of this article prompted HS to respond to IEEE Micro outlining his considered views on these subjects. Extensive discussions between the authors of this report were then organized resulting in the modified final response to IEEE Micro from HS and further contributions from the original authors of [1]. We believe that jointly these articles represent the “state of the art” in our views on these subjects and here they are unified in this report for reference purposes.

## **References**

- 1 Xia, F., Yakovlev, A., Clark, I., Shang, D., “Data communications in systems with heterogeneous timing”, IEEE Micro, 2002, 22, (6), pp 58-69.

PB 077B  
MBDA UK Ltd  
Six Hills Way  
Stevenage  
Herts SG1 2DA  
UK

10 September 2003

Group Managing Editor  
IEEE Micro  
10662 Los Vaqueros Circle  
PO Box 3014  
Los Alamitos  
CA 90720  
USA

Dear Janet,

**HETS Paper : IEEE Micro, 2002, 22, (6), pp 58-69**

A copy of this paper has only recently come to my attention. The paper includes a description of a scheme for classifying ACMs. My comments are confined solely to this aspect of the paper.

An important element of the paper is the description and use of a two-by-two matrix classification scheme which the authors claim as an original contribution ("Our new ACM classification system ..... p 61, lh column). This fails to recognise that such a scheme is well publicised and has existed (in the public domain) for over 10 years. There is also some technical incompleteness in the discussion of ACM properties in relation to this classification scheme.

I am concerned that readers of the paper will fail to see the maturity of the idea, and that the incomplete technical description will mean that they cannot appreciate its importance. Also, there are no suitable references which would allow a reader to see how the classification scheme has emerged in the context of a general system development method.

The note 'Asynchronous Communication Mechanisms' (attached) clarifies these points. It has been produced following discussions with the authors of the Micro paper.

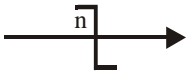
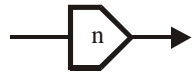
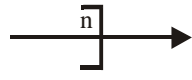
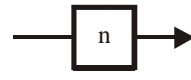
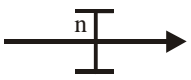
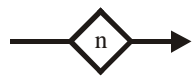

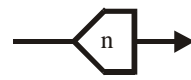
Yours sincerely,

Hugo Simpson  
Research Consultant and  
Visiting Professor at the University of Newcastle  
and Kingston University

email: hugo.simpson@mbda.co.uk

## ASYNCHRONOUS COMMUNICATION MECHANISMS

An ACM, as defined here, is a connector between two asynchronous (heterogeneously timed) processes, a writer and a reader, through which a series of data items can be passed. A recent paper [1] made use of a scheme for the classification of ACMs in terms of a two-by-two matrix of writing and reading properties, but it did not show clearly the essential similarity between this work and a pre-existing classification scheme [2-4], nor did it identify some important differences which are evident at the detailed level. The paper did contribute a new set of symbols for the four basic types of ACM, and it proposed a buffered form for ACMs which allow re-reading (but see below). This short note summarizes the similarities and differences between previous results and the work reported in the paper. Where necessary, points of technical detail have been clarified with the authors [1]. A comprehensive and formal description of the pre-existing classification scheme has since been published [5].

	DR RB NRR	NDR RNB RR
DW WNB OW	  <b>SIGNAL FAMILY</b> <b>OW-BB</b> $n \geq 0$ $n \geq 0$ <i>Event Data</i>	  <b>POOL</b> <b>OW-RR-BB</b> $n = 1$ $n \geq 1$ <i>Reference Data</i>
NDW WB NOW	  <b>CHANNEL FAMILY</b> <b>BB</b> $n \geq 0$ $n \geq 0$ <i>Message Data</i>	  <b>CONSTANT</b> <b>RR-BB</b> $n = 1$ $n \geq 0$ <i>Configuration Data</i>
Legend:	BB      Bounded Buffer DR      Destructive Reading Obligatory NDW      Non Destructive Writing Obligatory RNB      Reader Non Blocking WNB      Writer Non Blocking RR      Re-Reading Permitted OW      Over-Writing Permitted	n      Maximum Number of Items NDR      Non Destructive Reading Obligatory DW      Destructive Writing Permitted WB      Writer Blocking Possible RB      Reader Blocking Possible NRR      No Re-Reading Obligatory NOW      No Over-Writing Obligatory

### Interaction Protocols and Buffered ACMs

A single diagram can be used to describe the two-by-two matrix of ACM types. It gives symbols and names for each type; alternative reading properties are in the two columns, and alternative writing properties are in the two rows. The symbol on the left for each ACM type is a 'protocol', as described in [2-4]; the symbol on the right is a buffered ACM, as described in [1]. The protocol symbol expresses the rules for an interaction in terms of whether intermediate items in the ACM are destroyed or not by writing and reading. The buffered ACM symbol expresses whether over-writing or re-reading, or both, is permitted. The terms in bold italics describe the operational purpose envisaged for each ACM type.

The *capacity* 'n' of an ACM is the maximum number of items it can hold when neither process is engaged in an access operation. Data items currently being written [1-4], and data

items currently being destructively read [2-4], or read or re-read [1], are regarded as not present, but nevertheless space must be allowed for them in any implementation, to cover the assembly of an item during writing and the extraction of an item during reading. For example, the physical buffer size needed for a member of the channel family is  $n + 2$ . Where writing is destructive (over-writing permitted) the maintenance of coherence becomes an issue, and extra space may be needed for this.

Writing and reading operations, within the constraint imposed by the capacity, determine the current *contents* of the ACM:

- A non destructive write [2-4], or a write [1], increases the number of items by one, at the end of the write.
- A destructive write [2-4], or an over-write [1], leaves the number of items unchanged.
- A destructive read [2-4], or a read [1], reduces the number of items by one, at the start of the read.
- A non destructive read [2-4], or a re-read [1], leaves the number of items unchanged.

The protocols [2-4] and buffered ACMs [1] have slightly different conventions for determining the notional contents for non destructive reading (re-reading permitted). When the ACM contains a single item, the non destructive read [2-4] leaves this item undisturbed, and the contents remain at one item. However, a read [1] conceptually removes this item and reduces the contents to zero, subsequent access to this undestroyed (and still available) item being gained by a re-read.

The protocols and buffered ACMs have identical synchronisation properties:

- DR, NRR. Data must exist for reading to start.
- NDW, NOW. Space must exist for writing to finish.
- NDR, RR. No constraint on reading.
- DW, OW. No constraint on writing.
- RNB, WB, WNB, RB. The blocking properties follow directly from the above.

Consider first those ACMs where reading is destructive (no re-reading obligatory). The Channel Family comprises: Bounded Buffer ( $n > 1$ ); Channel ( $n = 1$ ); Rendezvous ( $n = 0$ ). The Signal Family comprises: Overwriting Buffer ( $n > 1$ ); Signal ( $n = 1$ ); Flash Data ( $n = 0$ ). These precisely match the buffered ACM types. Particularly interesting are the Rendezvous and Flash Data, which are highly synchronous from the operational viewpoint, but which can be implemented in a fully asynchronous execution environment.

Now consider the protocol form of the ACMs where reading is non destructive. It can be seen at once that, if the non destructive property is to be evident over all time, then the ACM must have the capacity to hold at least one item, and it must display an initial value.

Therefore  $n \geq 1$  for the Pool and Constant on the right of the matrix. When  $n > 1$ , buffering is provided, the semantics being modified so that, when there is more than one item in the buffer, a destructive read is applied to the oldest item. The non destructive reading property, which always holds for a single remaining item, implies that the value to be obtained is acceptable even when there is no new data since the last read. In between reads the writer may supply a 'queue' of new values, each better (fresher) than the last. The reader should not have to go through the queue of values, discarding or processing them one by one, but should be able to get straight to the latest (freshest) value.

There is an additional problem in the case of the 'buffered Constant', where any change in its value would be motivated by the writer but would only become evident within the system when it is accessed by the reader. Thus the writer could be frustrated in its role to supply new

up-to-date values over intervals when the reader does not access the ACM. One can conclude that buffering does not sit easily with the non destructive reading property. However, the unbuffered  $n = 1$  protocols with non destructive reading both provide important system functions. There is no point in connecting a writer to a Constant, whose value remains stable over the lifetime of a system instantiation.

Finally consider the buffered ACMs where re-reading is permitted. It makes sense that  $n \geq 1$  for the OW-RR-BB mechanism, as  $n = 0$  would mean that the writer would have nowhere to leave the latest value whilst moving on to generate a further value. However the RR-BB mechanism has a valid form of operation even when  $n = 0$ . An  $n = 0$  RR-BB mechanism will undergo an initial write and read, as do all the mechanisms [5]. Operation will start such that reads occurring before the first write are 're-reads' of the initial value. The first and subsequent writes will be held at the end of the write since they are unable to set  $n = 1$ . The next read to start will pick up the new data and will free the writer. This (pleasingly) works precisely like a writer for a Rendezvous. The buffered re-reading ACMs for  $n > 1$  (OW-RR-BB) and  $n > 0$  (RR-BB) are technically interesting, but they experience the same difficulties as the buffered non destructive reading protocols (see previous two paragraphs). It is not clear, from the operational viewpoint, what purpose they would serve.

The two-by-two ACM classification scheme is an important basic concept in a now mature system development approach [2-4] based on the use of explicit passive connectors for interaction between processes. The ideas apply equally to software and hardware architectures. When applied to system-on-chip applications, these ACMs can provide interaction mechanisms which have no need for global synchrony, and which in principle are free from any form of arbitration or mutual exclusion.

## References

1. Xia F, Yakovlev A V, Clark I G and Shang D. Data Communications in Systems with Heterogeneous Timing. IEEE Micro, 2002, 22, (6), pp 58-69.
2. Simpson H R. Layered Architecture(s) : Principles and Practice in Concurrent and Distributed Systems. IEEE Proceedings of the International Conference and Workshop on Engineering of Computer-Based Systems, Monterey, March 1997, pp 312-320.
3. Simpson H R. Architecture for Computer Based Systems. IEEE Proceedings of Tutorial and Workshop on Engineering of Computer-Based Systems, Stockholm, May 1994, pp 70-82.
4. Simpson H R. Real Time Networks in Configurable Distributed Systems. IEE Proceedings of the International Workshop on Configurable Distributed Systems, London, March 1992, pp 45-59.
5. Simpson H R. Protocols for Process Interaction. IEE Proceedings on Computers and Digital Techniques, 2003, 150, (3), pp 157-182.

# More thoughts on Asynchronous Communication Mechanisms

F. Xia, I. Clark, A. Yakovlev, D. Shang

First of all, our thanks go to Hugo Simpson for his keen interest in our work and the enlightening discussions in his article [1], which help to clarify many important points with regard to ACM classification and high-level specification. We would also like to thank Hugo Simpson and Eric Campbell for extensive and helpful discussions.

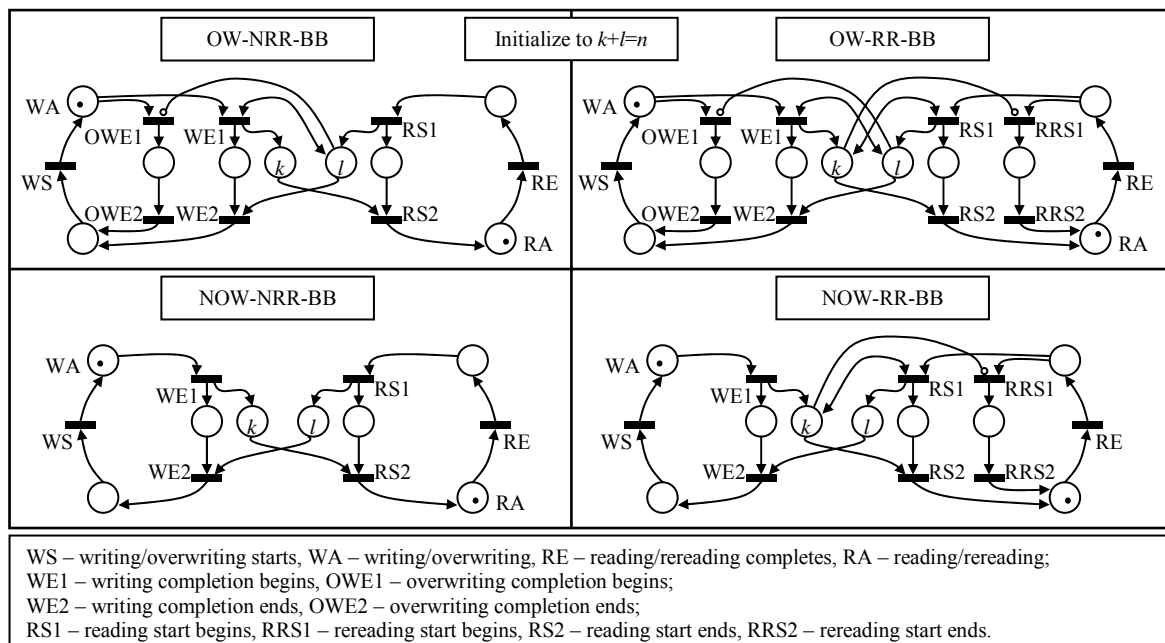
It was not our intention to claim the use of a 2x2 matrix in classifying ACMs as our original contribution. We would like to bring to the readers' attention paragraph 4, lh column, page 60, of our original paper [2], which partially reads: "Simpson proposed a more general classification system for ACMs, ... Here we further develop this classification ...". However, we apologize for not making the relationship between our work and Simpson's clearer, perhaps in the fashion of reference #10 of [2] and with better references. We hope this short note, in addition to [1], will help to redress that.

Simpson's classification system, using the destructive-ness of writing and reading as the differentiator, naturally results in four protocols (ACMs) which have distinct synchronization relationships between the writer and reader processes (whether the writer and reader may be required to wait). The "further development" mentioned above uses the number of unread data items in an ACM as the data state, and differentiates ACMs based on whether overwriting and rereading are permitted rather than whether writing and reading are destructive. This change preserves the important synchronization relationships, but allows us to obtain different ACMs in the right-hand column of the 2x2 matrix, as well as to extend them to other than just  $n=1$  (cf. Figure 3, top of page 160 of reference #5 of [1], which shows that traditionally, non-destructive reading protocols only allow  $n=1$ ). Some operational differences between the two approaches are highlighted below.

We substantially agree with most of the technical details brought up by Simpson. Moreover, we would also like to make a few additional technical comments to further clarify our views. In order to enhance comparative readability, we use the terminology from [1] rather than [2]. The rest of this note should, therefore, be read in conjunction with [1].

We use Petri nets as the main symbolic tool in classification, specification, implementation and verification. With this background, we use the following Petri net descriptions of ACMs, some of which have appeared in earlier work in various forms (e.g. reference #10 of [2]), to clarify our views.

Figure 1 includes Petri net models of maximally asynchronous versions of the four ACM types. These models describe how the data state of an ACM is modified at the start of a reader access and the completion of a writer access. The inhibitor arcs are used to specify clearly when rereading and overwriting may not happen. The reader and writer processes in these models do not share common transitions. This reflects the desire for maximal quantitative asynchrony once the qualitative temporal requirements for each type are satisfied. It is entirely possible for implementations to demonstrate reduced quantitative asynchrony while still adhering to the qualitative temporal requirements of a particular ACM type (cf. Figures 5 and 8 in [2] and the discussion on the many versions of the  $n=0$  NOW-NRR-BB, called "rendezvous", in reference #5 of [1]).



**Figure 1** Petri net models of maximally asynchronous ACM types.

The case of the  $n=0$  OW-RR-BB in Figure 1 is particularly interesting. With the correct initialization ( $k=l=0$ ), the reading and writing transitions will never fire and all writer accesses are overwriting and all reader accesses are



rereading. This means that the maximally asynchronous  $n=0$  OW-RR-BB behaves like the Constant in Simpson's classification, in that no new data written by the writer can ever replace the initial value which remains permanently available to the reader.

Figure 1 demonstrates that for ACMs, a larger  $n$  allows a higher degree of asynchrony (quantitatively) between the writer and reader processes. If the buffer is arranged to be FIFO (not specified at the level of classification in our case), data transit can be smoothed, which is indeed one of the primary reasons for FIFO buffering. As with all buffering schemes (such as in the NRR ACM types traditionally), gains in data transit smoothness are offset by necessary costs in latency. We do not agree that cases with  $n>0$  for the NOW-RR-BB and  $n\neq 1$  for the OW-RR-BB seem to have no practical value. This observation is based on the view that any new item of data generated by the writer should supersede all previous items generated. This desire is naturally served by OW ACMs with  $n\leq 1$ . We have therefore extended the range to include the buffered OW-RR-BB and the NOW-RR-BB types (not present in the previously defined protocols) as we consider it to be too early to predict the full range of ACM applications, and there may be trade-offs between latency and continuity which favour these types (e.g. in such applications as networking).

An example difference resulting from modifying the classification differentiator from the destructive-ness of reading and writing to whether overwriting and rereading are permitted, as mentioned in [2], is the  $n=1$  case of NOW-RR-BB (called "Message" in [2]), which is semantically different from Constant. For the Constant, because neither writing nor reading can be destructive, the initial item of data in the ACM will be available for rereading by the reader throughout the life of the system, so there is no practical value in connecting a writer process. For the  $n=1$  NOW-RR-BB, because reading accesses update the data state to unread, subsequent writing accesses may happen. Only when a writer is not connected on purpose would it operate in the same way as the Constant.

Operationally, the Pool and the  $n=1$  OW-RR-BB can be considered equivalent, in both synchronization and data passing properties. However, the semantics of the destructive-ness of reading and writing deliberately does not distinguish rereading from reading and overwriting from writing. For the Pool, writing is always destructive and reading never destructive. This emphasizes that for the Pool's envisaged application (reference data), whether a data item acquired by the reader was previously acquired, and whether a writer data access destroys a previous item which has not yet been acquired by the reader, are not of interest. Our system based on whether overwriting and rereading are permitted, however, obliges us to distinguish between overwriting and writing as well as rereading and reading, because they are predicated by the data state and update the data state in different ways.

In [1], it is proposed that NDR protocols may be extended to  $n>1$  by the semantic modification (obligatory NDR to permitted NDR) such that if there is more than one item of data in the ACM at the beginning of a reader access, the oldest one is destroyed. NDR, however, remains obligatory for the last item in the ACM. If all items in the ACM have been read, there will be one last item of already read data not destroyed. If the writer then supplies a new item, the reader will still need to reread (and destroy) the older item before being able to access the new one. In an RR ACM the older item would have already been marked as read, thus the reader is immediately directed to the new item (Figure 1, inhibitor arcs preventing RRS). Our "further development" allows us to extend the RR ACMs to  $n>0$ , without having to modify the semantics.

## References

- 1 Simpson, H.R., "Asynchronous communication mechanisms", elsewhere in this issue.
- 2 Xia, F., Yakovlev, A., Clark, I., Shang, D., "Data communications in systems with heterogeneous timing", IEEE Micro, 2002, 22, (6), pp 58-69.