occam™ user group newsletter

No. 1 Summer 1984

CONTENTS

and occam are trade marks of the INMOS Group of Companies

CONTRIBUTIONS WANTED

The sucess of the Newsletter depends entirely on contributions from you, the users of and thinkers about occam and the transputer. We welcome short contributions on any aspect of occam and the transputer, such as applications experience, evaluations of the language and concept, useful short programs, notes on teaching, theoretical background....in fact anything you would like to communicate to other users or potential users. For inclusion in the next Newsletter, please send contributions to the Editor:

Dr Martin Bolton
Dept of Electrical and Electronic Engineering
University of Bristol
Queen's Building
University Walk
BRISTOL BS8 1TR

THE OCCAM PROGRAMMING SYSTEM

by Philip Mattos, INMOS

The OCCAM Programming System (OPS) is now fully available for the VAX under VMS, and offers the following features:

occam development system for VAX/VMS
integrated screen editor/checker
optimising VAX compiler

It will be upgraded during the rest of this year to also offer:

full occam....floating point, etc
separate compilation facilities
transputer development system
other hosts

The ops provides an effective development environment for the creation of occam programs for execution on VAX/VMS.

The editor/checker provides both the features required of a screen editor and the textfolding mechanisms to provide control of the hierarchy of a large program, together with the ability to syntax check the source.

Textfolding enables a section of the source - program or text - to be 'hidden' behind a commented special marker. Such a fold may itself contain folds, allowing the program text to be hidden in a hierarchical fashion. Viewing the content of a fold requires simply that the cursor be moved to the fold marker, and the 'open fold' key be pressed; the text of the fold is then unwrapped onto the screen. Creating a fold requires marking above and below the text by cursor

movement and hitting the 'create fold' key: the text is folded out of sight, and the marker shown. To provide a simple identification of the folded text, the marker may be commented.

The fold mechanism is used in several of the ops major operations; syntax checking is done on a fold (allowing partial checking of the source), and the equivalents of 'include' files and the facilities of separate compilation will be implemented using it. Text once folded may be placed in a separate file; conversely text in a separate file may be attached to a fold. This provides a simple mechanism for the sharing of source files, while allowing the user access to the text of the files wthin a natural context. Separate compilation will be achieved by the compilation of such 'outfolded' source files; compiling the main program will simply cause the separately compiled folds to be linked.

The source of the editor is provided to allow user-customising for special requirements and for tailoring to specific terminals. As supplied, the ops works with vt100 and TVI920 terminals.

The separate compiler, invoked to produce an executable file after leaving the editor/checker when no syntax errors remain, generates well-optimised code. Itself written in occam (as is the editor/checker), the compiler is an example of a large concurrent occam program, being organised as five major communicating processes.

Technical Summary

Editor/checker functions

The editor/checker functions use the terminal cursor keys and the numeric keypad. The functions currently are:

```
-- move the cursor as indicated
-- func
-- delete line
-- undelete line
-- open fold
-- close current fold
-- move cursor to left end of line
-- pick or put line (alternate)
-- copy current line and insert on line above
-- move cursor to right end of line
-- move screen down document
-- make last line of fold bottom line on screen
-- make current line centre of screen
-- move screen up document
-- make first line of fold top line of screen
-- write file away and exit editor
```

```
-- create fold
-- remove fold
-- syntax check current fold
-- move cursor to line containing syntax error,
   opening folds as necessary
```

OPS Products

VAX/VMS available now

First release will implement an integer subset of full occam, and omit the extensions of file folding to separate files and separate compilation.

Next release - available free to previous purchasers - will implement full occam, which will provide multilength arithmetic, structures and IEEE real arithmetic. In addition, th full folding mechanisms will be supplied.

VAX/Unix available Q4 1984

Same facilities as ops VAX/VMS

INMOS Workstation available Q4 1984

A 68000-based workstation with Winchester, floppy disk, half megabyte of memory and a terminal together with the ops software.

Related Products

OCCAM Evaluation Kit available now

A low cost introduction to occam, available for a number of small computers.

OCCAM Portable Compiler Kit available Q4 1984

The occam compiler with a code generator for an abstract machine for which the user writes an interpreter as the first stage in porting the compiler to hosts and/or operating systems not directly supported by INMOS.

OPS/IBM PC available Q1 1985

A portakit implementation of the ops on an IBM PC or PC XT, under MS-DOS.

Transputer Development System available Q4 1984

Software tools for IMS T424 support. Available as an upgrade to ops customers, when the cost of the ops is deducted from the tds price.

PROGRAM EXCHANGE

In this section we will give details of useful occam programs which are offered to others. The program exchange coordinator is Sue Peeling of RSRE, whose address is given below. Please send descriptions of programs you would like to make available to her.

1.Floating Point Arirhmetic

A VAX OPS package is provided for representing and arithmetically operating on floating point numbers in OCCAM. The numbers are stored to 6 significant figures in a binary code decimal format. Multiplication produces a 12 figure result which is rounded up to 6 figures. All other operations produce 6 figure results although only 4 significant figures are displayed. Monadic and dyadic minus are supported.

Apart from the standard arithmetic operations, shift, rounding (always up), normalisation, relational operations, integer to floating point conversion and input/output are provided.

The numbers are stored in byte arrays with two digits per byte. To increase the speed in multiplication and division a routine to convert to one digit per byte is used.

The range of floating point numbers supported is

-0.9999 * 10**-62 to 0.9999 * 10**62

Any attempt to go outside this range produces an error message.

All calculations use the full 6 figures which are stored.

Further details are available from:

Mrs S.M.Peeling,
R.S.R.E,
St. Andrews Road,
Malvern,
Worcs WR14 3PS

Tel: Malvern (06845) 2733 ext 2228

USE OF OCCAM FOR THE DESCRIPTION OF DIGITAL FILTERING HARDWARE

by J.W.Harrison (3rd year student)
Department of Electrical and Electronic Engineering
University of Bristol

This article examines briefly the use of occam in the description and simulation of digital filters. The language

is very useful in this application, and the principles could easily be extended to cover other design areas.

The description and design of a particular class of systolic architecture [1] was studied in detail. The architecture considered has a separate processing element associated with each filter weight. The advantages of this type of design include the use of an array of similar processing elements, which are regularly interconnected. Occam is well suited to this type of design since the replicated PAR construct can be used to create the processing elements. These can be connected easily by channels to represent the wiring which would be used in practice. Typically, the replicated processes would take in data and a partial sum of previous weighted data samples. The data would then be multiplied by a weight, and added to the partial sum. This result could then be sent to another stage in the array of processes. In this way, simple non-recursive filters can be simulated. The method can also be extended to form recursive filters, and a Fast Fourier Transform processor. Details of these structures can be found in Lynn [2], and [3].

One of the major advantages of using a structured language in this way is that the description can be extended to as low a level of detail as is required. In the examples above, the processing elements can be simulated more closely by modelling the behaviour of the actual devices which would be used, for example the multiplier circuits. The low level simulation can take into account such factors as output delay times, and input setup times. This can allow extensive debugging of the final design without the need to commit anything to hardware. The model can be used for an initial feasibility study right through to the final design, with simulation and evaluation at each stage.

It would seem to be relatively straightforward to build up a standard library of devices which are commonly used, and these could then be called up in parallel as required in particular projects. The only real problem encountered with the study undertaken was that of the tight synchronisation required between occam processes. This was overcome by using extensive buffering of signals, both on inputs to the procedures, and their outputs. The buffering procedures also allow error messages to be printed when a procedure produces a second output before the first has been read, or if an input changes several times before the clock to the circuit has latched it in.

The principal advantage of occam over a purely sequential structured language such as Pascal is that the program bears a greater resemblance to the actual hardware. Separate devices are represented by separate parallel processes. Wiring is represented directly by the channel interconnections. Thus occam provides a powerful and useful simulation and design aid in this application area.

REFERENCES

1 Kung H.T., 'Why systolic architectures ?', IEEE Computer , Vol 15 No 1, Jan 1982, p37-47.

2 Lynn P.A., An Antroduction to the Analysis and Processing of Ssignals, Macmillan Press, 1980.

3 Bergland G.D., 'Fast Fourier Transform hardware implementations - an overview', IEEE Trans. Audio and Electroacoustics, Vol AU-17, No 2, June 1969, p104-108.

A SIMPLE RANDOM NUMBER GENERATOR

submitted by Gordon Harp, RSRE

Random number generators have applications in program testing, data generation and games. A procedure to generate pseudo-random numbers is given using the technique of m-sequence shift registers.

A seed, ranum, is loaded into a 16 bit shift register with taps at bits 0,2,11 and 15 which are selected by ANDing with an appropriate constant. All selected bits are shifted to bit 0, exclusively-ORed and fed back into the most significant bit. The register is clocked to produce a new random number after 16 clock cycles. Latch-up is prevented by checking that the register does not get into a state of all zeros.

The sequence has a uniform distribution and repeats after (2**16)-1 calls.

For the majority of applications, requiring only random bytes, the least significant bits should be used by ANDing ranum with #FF.

Take care when typing the program!

```
PROC ranum(VAR ranum)=  --16 bit m-sequence generator
  IF
    ranum=0             --check that seed is not zero
      ranum:=#1357      --if so, set to default value
    TRUE
      SEQ i=[0 for 16]
        ranum:=(((((ranum/#1)<<15)><((ranum/#10)<<11))><
               ((ranum/#2000)<<2))><(ranum/#8000))/(ranum>>1):

-- To show the use of PROC ranum
--
VAR random.number
SEQ
  random.number:=#1111    --seed value
  ..
  ..
  random(random.number)
  random.number:=random.number/#FF   --to give 8 bit value
```

OCCAM USER GROUP TECHNICAL MEETING

21st September 1984

The OCCAM User Group are arranging a one-day technical meeting to be held at The Watershed, Bristol on the 21st September. Any member who would like to attend this meeting should get in touch with Michael Poole at the INMOS Bristol Office.

Provisional List of Contributors

Bill Roscoe, Oxford Univ PRG
"Formal Semantics of OCCAM"

Roland Backhouse, Essex University
"Experience of teaching occam ~~to students~~"

Mike Reeve, Imperial College
"OCCAM in the ALICE Project"

John Ainscough, Brunel University
"An OCCAM Compiler"

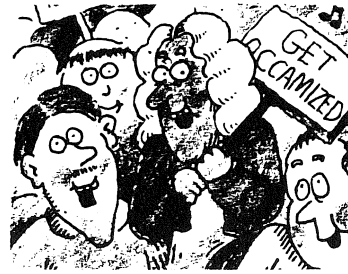
Sue Peeling, RSRE.
"Experiments with OCCAM for Sorting"

Don Fay, Queen's Univ, Belfast
- to be decided -

Someone from Standard Telephone Labs
- to be decided -

Richard Bornat, Queen Mary College
"A Proposal for Output Alternatives in OCCAM"

David May, INMOS
"OCCAM, the Transputer, and the Future"



SORTING ON THE TRANSPUTER

by S.M.Peeling, RSRE

This work arose from the desire to asses the suitability of using the transputer to calculate the median, or middle, value of a set of numbers. Various specialised methods were considered but the most appropriate solution seemed to be the adaptation of a general sorting routine. To this end five general sorts were considered and occam programs written to implement each one. Included in these programs were statements which counted how many 10ns cycles were involved in each operation. In this way it was possible to run each program on sets of random numbers and compare the timings obtained.

The five sorts are described below. In each case it is assumed that N numbers are to be sorted with the largest at the top (right hand end). Some of the methods allow advantage to be taken of the fact that the median is the middle value in the sequence, and hence it is only necessary for half the numbers to be sorted. Such methods are indicated.

1.Bubble Sort

This is one of the simplest sorts and involves "bubbling" the highest (or lowest) number to the top of the group. Adjacent numbers are compared and if the first is greater than the second they are swapped. This is repeated throughout the entire sequence, i.e. first and second, second and third, third and fourth etc are compared. This can be repeated until all N numbers are sorted at a cost of

$$(n-1) + (N-2) + \ldots\ldots\ldots + 1$$

comparisons and up to the same number of exchanges. It is only necessary to sort half the numbers in the median calculation.

2.Q Sort

This is also called the linear insertion sort. This is another easily implemented sort which also utilises the fact that only half the sorted numbers are required. Here the numbers are fed in one at a time. The first number is assumed to be the largest and so occupies the top position. The second number is compared to the first and if it is smaller it occupies the second position, otherwise the first number moves down one position and the second number replaces it at the top. Each succeeding number is compared with those already in position and shifts performed, as necessary, to insert it in the correct position. It is not, of course, necessary to fill the lower half of the group although all the numbers are input. As an example consider the input sequence 5 9 3 7 which results in:

i)	5		
ii)	9 5	1 comparison	+ 1 shift
iii)	9 5 3	2 "	+ 0 shift
iv)	9 7 5 3	2 "	+ 2 shift

3.Shell Sort

This is also called the diminishing increment sort since it sorts pairs of numbers initially some distance d apart then successively halves d until it becomes zero. It is claimed to be more efficient than Bubble Sort but no more difficult to implement. There seemed to be several different versions of the routine and also several ways of determining d. The comparisons tend to overlap and hence hence propagate the exchanges.

4.Quicksort

This is also calles the partition exchange sort since it relies on partitioning the numbers so that a certain element v is in it final position. All elements above v are greater than it whilst all below are less than v, although nothing can be said about the order of these subgroups. If necessary, these subgroups can also be partitioned although this application did not do so. It is hoped that v will be near the middle of the group and thus only a small amount of sorting will be necessary to produce the median. Again there seemed to be several different versions of this routine available, of varying complexity.

5.Straight Selection Sort

This is probably the simplest sort to implement and is similar to Bubble Sort. It involves finding the largest number and swapping it with the one in the Nth (top) position. This is then repeated over the lower (N-1) numbers and the largest swapped into the (N-1)th position. It has the same number of comparisons as Bubble Sort but fewer shifts. Again it is only necessary to sort half the numbers.

The average timings over 6 sets of 9 random numbers are shown in Table 1.

Table 1

	Time (microseconds)
Quicksort	40.10
Straight Selection Sort	51.22
Shell Sort	72.36
Q Sort	76.69
Bubble Sort	79.21

Although Quicksort seems to be the best method it is worth noting that the algorithm is the most complex to implement and the timings ranged from 30 to 66 microseconds. Straight Selection Sort is worthy of consideration since it is an extremely simple algorithm and there was little variation over all the timings.

Further details of this work, including occam code listings, can be obtained fron the author, whose address appears in the Program Exchange section of this newsletter.

BIBLIOGRAPHIES

A regular feature of the newsletter will be a bibliography section containing lists of items on or related to occam and the transputer. The first one, on papers related to Communicating Sequential Processes, was provided by Geraint Jones of the Programming Research Group at Oxford, and the second, containing papers about occam and the transputer, was provided by INMOS, with additions by the Editor.

These will be updated in every newsletter. If you know of any item which has been missed, please inform the Editor and it will be included.

1. A CSP Bibliography

K.R.Apt, "Formal justification of a proof system for communicating sequential processes," Journal of the ACM , Vol. 30, No. 1, Jan 1983, pp 197-216.

K.R.Apt, N. Francez, W.P.de Roever, "A proof system for communicating sequential processes," ACM Transactions on Programming Languages and Systems , Vol. 2, No. 3, July 1980, pp 359-385.

S.D.Brookes, A Model for Communicating Sequential Processes , Oxford University Programming Research Group, Technical Monograph PRG-35, 1983.

S.D.Brookes, "On the relationship of CCS and CSP," Proceeding of the 10th Colloquium on Automata, Languages and Programming , Springer Lecture Notes in Computer Science, No. 154, 1983, pp 83-96.

S.D.Brookes, C.A.R.Hoare, A.W.Roscoe. "A Theory of Communicating Sequential Processes," Journal of the ACM , Vol. 31, No. 3, July 1984.

S.D.Brookes, A.W.Roscoe, An Improved Failures Model for Communicating Sequential Processes , Carnegie-Mellon University Technical Report, to appear, 1984.

E.C.R.Hehner, C.A.R.Hoare, "A more complete model of communicating processes," Theoretical Computer Science , Vol. 26, 1983, pp 105-120.

C.A.R.Hoare, "Communicating Sequential Processes," Communications of the ACM , Vol. 21, No. 8, Aug 1978, pp 666-676.

C.A.R.Hoare, A Model for Communicating Sequential Processes , Oxford University Programming Research Group, Technical Monograph PRG-22, 1981.

C.A.R.Hoare, "A Calculus of Total Correctness for Communi-

cating Processes," Science of Computer Programming , Vol. 1, 1981, pp 49-72.

C.A.R.Hoare, Specifications, Programs and Implementations , Oxford University Programming Research Group, Technical Monograph PRG-29, 1982.

C.A.R.Hoare, Notes on Communicating Sequential Processes , Oxford University Programming Research Group, Technical Monograph PRG-33, 1983.

C.A.R.Hoare, A.W.Roscoe, The Laws of Programming , Oxford University Programming Research Group, to appear, 1984.

J.R.Kennaway, C.A.R.Hoare, "A theory of nondeterminism," Proceedings of the 7th Colloquium on Automata, Languages and Programming , Springer Lecture Notes in Computer Science, No. 85, 1980, pp 338-350.

S.S.Kuo, M.H.Linck, S.Sadaat, A Guide to Communicating Sequential Processes , Oxford University Programming Research Group, Technical Monograph PRG-14, 1979.

R.de Nicola, "A complete set of axioms for a theory of communicating sequential processes," Proceedings of the International Conference on Foundations of Computation Theory , Springer Lecture Notes on Computer Science, to appear, 1983.

E.-R.Olderog, C.A.R.Hoare, Specification-Oriented Semantics for Communicating Processes , Oxford University Programming Research Group, Technical Monograph PRG-37, 1984.

G.D.Plotkin, "An operational semantics for CSP," Formal Description of Programming Concepts, II , North-Holland, 1983, pp 199-223.

A.W.Roscoe, A Mathematical Theory of Communicating Sequential Processes , DPhil thesis, Oxford, 1982.

A.W.Roscoe Denotational Semantics for occam , Oxford University Programming Research Group, to appear, 1984.

W.C.Rounds, S.D.Brookes, "Possible futures, acceptances, refusals and communicating processes," Proceedings of the 22nd IEEE Symposium on Foundations of Computer Science , Nashville, Tenessee, 1981.

N.Soundararajan, O.-J.Dahl, Partial Correctness Semantics of Communicating Sequential Proceses , Institute of Informatics, University of Oslo, Resaerch Report No. 66, 1982.

Zhou Chaochen, The Consistency of the Calculus of Total Correctness for Communicating Sequential Processes , Oxford University Programming Research Group, Technical Monograph

PRG-26, 1982.

Zhou Chaochen, C.A.R.Hoare, "Partial correctness of communicating processes," Proceeding of the 2nd International Conference on Distributed Computing Systems , Paris, 1981.

2. OCCAM and the Transputer Bibliography

OCCAM papers authored outside INMOS

S.Fawcett, "OCCAM talks on parallel lines," Computing , Dec 2, 1982, p 19.

A.N.Godwin, "Simulation and global time in occam," presented at System Science VIII , Wroclaw, Poland, 1983.

Dick Pountain, "OCCAM's curtain raiser," Soft , June 1983, pp 57-59.

M.Banks, "The coming of concurrency," Systems International , June 1983, pp 73,74.

Max Schindler, "Real-time languages speak to control applications," Electronic Design , July 21, 1983, pp 105-120. (Uses occam for representing logic gates and control charts, in review of several languages)

Graham R. Perkins, Letter to Editor, ACM SIGPLAN Notices Nov 1983, pp 19,20. (Comments on the language)

A.Dixon, "occam - a concurrent programming language," CCTA News , Nov 13, 1983, pp 20,21. (Review)

Don Fay, "Working with occam: a program for generating display images," Microprocessors and Microsystems , Vol. 8, No. 1, Jan/Feb 1984, pp 3-15. (3D graphics on an Apple OEK!)

Jon M Kerridge, Dan Simpson, "Three solutions for a robot arm controller using Pascal-Plus, occam, and Edison," Software Practice and Experience , Vol. 14, No. 1, Jan 1984, pp 3-15.

B.Jane Curry, "Language based architecture eases system design. III," Computer Design , Jan 1984, pp 127-136. (Modelling hardware communucation and multitasking)

OCCAM papers authored by INMOS

Imnos Ltd., OCCAM Programming Manual , Prentice-Hall International, 1984.

B. Lee Jones, "OCCAM - a process oriented language for distributed processing," Proceeding of the Digital Equipment

Users Society , St. Louis, Missouri, May 1983, pp 305-310. (Uses matrix multiplication example)

David May, "OCCAM," ACM SIGPLAN Notices , Vol. 18, No. 4, April 1983, pp 69-79. (Language description with system example)

David May, "Large languages versus small languages," presented at IFIP Panel, Sept 19, 1983.

David May, "OCCAM," IEE Colloquium on Software Tools for Hardware Design , Digest No. 1983/98, 7 Dec 1983, pp 5/1-5/5.

David May, Richard Taylor, "OCCAM," Microprocessors and Microsystems , Vol. 8, No. 2, March/April 1984. (Includes examples of program transformation)

Richard Taylor, Pete Wilson, "Process-oriented language meets demand of distributed processing," Electronics , Vol. 55, No. 24, Nov 30, 1982, pp 89-95.
(Less formal version of Sigplan article)

Pete Wilson, "Programming System builds multiprocessor software," Electronic Design , July 21, 1983, pp 129-134. (Walks through the development of an occam program)

Pete Wilson, "OCCAM architecture eases system design - Part I," Computer Design , Nov 1983, pp 107-115. (OCCAM used as a system description language)

Pete Wilson, "Language based architecture eases system design - Part II," Computer Design , Dec 1983, pp 109-120. (OCCAM used for simple graphics)

Papers about the Transputer authored outside INMOS

F. Warren Burton, M. Roland Sleep, "Executing functional Programs on a virtual tree of processors," Proc. ACM/MIT Conference on Functional Languages and Computer Architectures , Oct 1981.

Ehud Shapiro, Lecture notes on "The Bagel: a systolic concurrent Prolog machine" ICOT Research Center Technical Memorandum TM-0031, Nov 1983. (Proposes transputer implementatin of concurrent Prolog)

T.Palmer, "Cagey INMOS reveals all," Infomatics , Vol. 4, No. 12, Dec 1983, pp 40-43,61.

M.Persson, "Transputer and occam: English microcircuit building block for the computers of the future," Industriell Datateknik (Sweden), Vol. 4, No. 1, Jan 1984, pp 39-43. (In Swedish)

T.Durham, "INMOS: a final frontier?" Computing , Jan 19, 1984, p 34.

Max Schindler, "Multiprocessing systems embrace both new and conventional architectures," Electronic Design , March 22, 1984, pp 77-130.

R.W.Coles, "The transputer - a component for the fifth generation," Practical Electronics , April 1984, pp 26-31.

P.Petre, "A computer chip with a mind of its own," Fortune , Vol. 109, No. 10, May 14, 1984, p 74.

Papers about the Transputer authored by INMOS

INMOS Ltd, IMS Transputer Advance Information , Nov 1983.

I.M.Barron, "The transputer," Mini/Micro West , Session Record 2: 16/32-Bit Microprocessor Architectures, Nov 8-11, 1983, pp 2/5 1-8. (Introduction to IMS T424)

Peter Cavill, "Transputer systems," Mini/Micro West , Sesion Record 19: System Design with 16/32-Bit Microprocessors, Nov 8-11, 1983, pp 19/5 1-6. (Introduction to transputer systems)

Iann Barron, Peter Cavill, David May, Pete Wilson, "Transputer does 5 or more MIPS even when not used in parallel," Electronics , Vol. 56, No. 23, Nov 17, 1983, pp 109-115. (Introductory article)

Stephen Brain, "The transputer - exploiting the opportunity of VLSI," Electronic Product Design , Dec 1983, pp 41-44.

Peter Eckelmann, "The transputer: a microcomputer concept for a high processing capability," Elektronik (Germany), Vol. 32, No. 24, Dec 2, 1983, pp 51-55. (In German)

Stephen Brain, "Applying the transputer," Electronic Product Design , Jan 1984, pp 43-48.

P.Eckelmann, "Architecture and use of the transputer," Elektronik (Germany), Vol. 33, No. 4, Feb 24, 1984, pp 59-65. (In German)

David May, Roger Shepherd, "OCCAM and the transputer," Preprints of the IFIP Workshop (WG. 10) on Hardware Supported Implementation of Concurrent Languages in Distributed Systems , Univ. of Bristol, March 26-28, 1984.

Useful background material

I.M.Barron, "Future developments in computer hardware and architecture," Computer Design: International State of the

Art Report , C.Boon, ed., Infotech, 1974, pp 501-512.

I.M Barron, "The decline and fall of the computer," Minicomputer Forum , (London, Feb 11-13, 1975), Online, 1975, pp 17-29.

I.M.Barron, "Intercommunication within distributed systems," Small Systems Software , Vol. 1, No. 2, 1976, pp 6-10. (Also in: Distributed Systems: International State of the Art Report , J.P.Spencer, Ed., Infotech, 1976)

A.M.Walsby, "Off the buses," Systems , June 1976, pp 24,25. ("Many of the ideas here were gleaned from Iann Barron")

I.M.Barron, "The Microcomputer and its Consequences," Digital Systems Design , Proceedings of the joint Newcastle/IBM Seminar, (Sept 6-9, 1977), pp 19-27.

I.M.Barron, "The transputer," The Microprocessor and its Applications , Cambridge Univ. Press, 1978, pp 343-357.

M.A.Jackson, "Information systems: Modelling, Sequencing and trasformations," Proc. of the 3rd International Conference on Software Engineering , 1978, pp 72-81.

I.M.Barron, "The future of the microprocessor," Microelectronics , Vol. 8, No. 4, June 1978, pp 32-36.

I.M.Barron, "The future of computer technology," Information Technology 78 , (Jerusalem, Aug 6-9, 1978), North-Holland, 1978, pp 125-132.

Iann Barron, Ray Curnow, The Future with Microelectronics The Open University Press, 1979.

C.A.R.Hoare, "The emperor's old clothes," (The 1980 ACM Turing Award Lecture), Communications of the ACM , Vol. 24, No. 2, Feb 1981, pp 75-83.

I.M.Barron, "Architecture Oriented Objects," Proceedings, Conference on Advanced Research in VLSI , P.Penfield, Jr., Ed., (MIT, Jan 25-27, 1982), Artech House, 1981, p 67. ("In the most recent period, the rococo, microprocessors are incorporating all the computer science ideas that never got into real computers.")

Richard Taylor, "Introduction to VLSI," SERC/DOI IKBS Architecture Study, Workshop 1 , Jan 6,7, 1983.

Butler W.Lampson, "Hints for computer system design," Proc. of the 9th ACM Symposium on Operating System Principles , Oct 1983, pp 33-48.

C.A.R.Hoare, "Programs are predicates," Meeting of the Royal Society on Mathematical Logic and Programming Languages , Feb 15,16, 1984.