occam® user group · newsletter

for all users of occam and the transputer

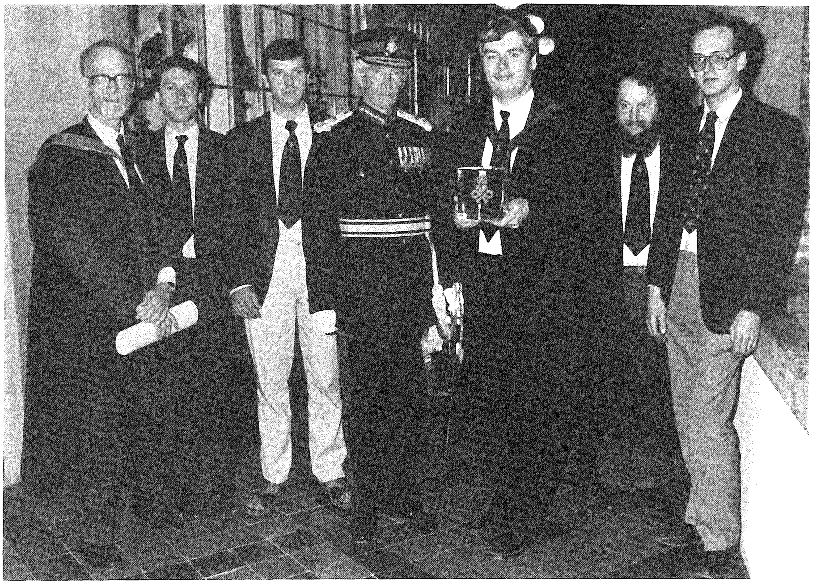
·O·U·G·

Nº 13 July 1990

Contents

EDITORIAL 2
Contributions to the newsletter 3
Proceedings of user group meetings 4
Newsletter back issues 5
FORTHCOMING 5
Trollius workshop 5
Thirteenth occam user group technical meeting 6
Edinburgh parallel computing centre first annual seminar 7
Symbols versus neurons? 9
First Nordic transputer seminar 11
North American transputer users group Fall meeting 14
Transputing 1991 14

continued on back cover



Bill Roscoe receives the symbol of a Queen's Award for Technological Achievement from the Lord Lieutenant of Oxfordshire; with from left to right Tony Hoare, Tony Cox, Geoff Barrett, Michael Goldmsith, and David Shepherd (see page 74)

EDITORIAL

The fellow in the fancy dress on the front cover – the one holding the glass brick – is Bill Roscoe, receiving on behalf of Oxford University Computing Laboratory a Queen's Award for Technological Achievement from the Lord Lieutenant of Oxfordshire, representing the Queen. This award and a matching one to Inmos are a pat on the back for a team effort that yielded practical benefits: friends of the formal method will welcome these awards as recognition of the synergy of sound engineering practice and sound mathematical research. We should do what we can to use this sort of publicity to draw attention to the benefits of 'doing it properly'. In case you do not know it already, you will find the story behind the award on page 74.

If that seems to you like a bit of English parochialism, you will also find in this newsletter the usual spread of contributions. In keeping with the new geography of the nineties, let me offer you a report on the things *parallel* in Bulgaria (page 20) as well as Japan (page 30); and contributions from as far apart as Cornell and Moscow, Silicon Valley and Hong Kong, Exeter and Tokyo, Buenos Aires and 'somewhere in the Baltic'.

Looming ever larger on the horizon is the joint meeting of user groups in California next spring – *Transputing 1991* – details of the call for papers for which you will find on page 14. Your ideas are also solicited for the workshops and tutorials that will happen at the same time, and indeed any good ideas you have to make this meeting the success it deserves to be would be welcomed by the committee. Who knows, we may even see the H1 there.

Speaking for the metropolitan *occam user group* for a moment, let me draw your attention to the intention to form a new special interest group dealing with development of the occam language (page 38). There is a lot going on, driven partly by the need for a higher-level means of expressing parallel programs than we now have, and partly of course by the potential capabilities of new generations of transputers and similar devices. If enough people are interested, the new group will meet at the York meeting (page 6).

Recent hardware developments seems to have been going all-out for performance, with a competition between the like of transputer boards with i860 on them (for example see page 83), and some spectacularly 'solid' lumps of very high-performance store-and-transputer on tiny TRAMs (see for example page 80). If you are looking for an affordable way to play with transputers – 'play' is pronounced 'teach' where I come from – the star of the show at the Exeter meeting, and a few other meetings too no doubt, has to be the *transputer education kit* from CSA (see pages 34 and 72).

Name of a dog: 'Rover'

Those of you who were at the last meeting of the occam user group, in Exeter, may recall Peter Welch publicly raising the matter of a potential change of name for the group. At the risk of being confused with the British 'Liberal Democratic Party' (is it really called that still?) allow me to expand on the subject. ('Could you compose something...?' he said; why it ends up being me I do not know.)

There is apparently a perception 'out there' that anything called the *occam user group* cannot have anything to do with transputers; perhaps even that it is a club for rather odd European academic computing types. This does not match the 'forum for all users of the transputer' slogan in the chairman's address (Newsletter Nº 10), nor the intention that occam should encompass 'not only the transputer but also its best applications, languages, support tools and methods.' We do need to make it clear that we are a forum for *all* users of the transputer – manufacturers, system designers, researchers through to end-users.

When all else fails, try changing the name. Peter suggests that we consider as possibilities:

▹ occam user group (original flavour);
▹ transputer user group;
▹ occam and transputer user group;
▹ transputer and occam user group;
▹ association for transputing machinery...

or any other good ones that come up. It is intended to put this to the membership at the forthcoming York meeting. What's in a name? Quite a lot perhaps.

Geraint Jones, 10 June 1990

CONTRIBUTIONS TO THE NEWSLETTER

Please contribute announcements, articles, letters about anything that looks as though it belongs in your *Newsletter*. In particular we welcome letters, short articles or news about work being done with occam or transputers; calls for, discussion of, and reports on meetings of the group or related societies; ideas for new ways the group could help its members, or better ways of organizing what we do; details of material published elsewhere in books and journals; information about new products and courses.

Life would be easiest for the editor if you were able to submit material of longer contributions by electronic mail to `oug-news@uk.ac.oxford.prg`; or to send either unformatted ASCII files (on an IBM PC compatible floppy disk, or some other medium by negotiation) or clean camera-ready copy to the editor at the address below.

Camera-ready copy should be arranged not to look out of place when its linear dimensions are reduced to about 70%, i.e. from A4 originals to the A5 page size of this booklet. (This means: make sure your type is not too small!) Pictures are welcome as black-and-white prints, and will be subjected to the same reduction in size.

Copy for the next edition must arrive by *23rd November 1989.*

Geraint Jones
Programming Research Group
11 Keble Road
Oxford OX1 3QD
United Kingdom

Tel: +44 865 273851
Fax: +44 865 273839
oug-news@uk.ac.oxford.prg

PROCEEDINGS OF USER GROUP MEETINGS

Proceedings of the meetings of both the *occam user group* and the *North American transputer user group* are now published for the groups by the IOS Press, and should be ordered directly from the publishers.

The following volumes are available: in the Occam User Group Progress Reports series (prices in Dutch guilders/US dollar prices in the USA and Canada only)

- ▷ OUG–7, 14–16 September 1987, Grenoble, *Parallel programming of transputer based machines*, ed. Traian Muntean (pp. x+480, fl.230/$110).
- ▷ OUG–8, 27–29 March 1988, Sheffield, *Developments using occam*, ed. Jon Kerridge (pp. vii+214, fl.92/$50).
- ▷ OUG–9, 19–21 September 1988, Southampton, *Occam and the transputer – research and applications*, ed. Charlie Askew (pp. vii+176, fl.92/$50).
- ▷ OUG–10, 3–5 April 1989, Enschede, *Applying transputer based parallel machines*, ed. André Bakkers (pp. viii+318, fl.130/$65).
- ▷ OUG–11, 25–26 September 1989, Edinburgh, *Developing transputer applications*, ed. John Wexler (pp. x+206, fl.115/$55).
- ▷ OUG–12, 2–4 April 1990, Exeter, *Tools and techniques for transputer applications*, ed. Stephen J. Turner (pp. vii+244, fl.130/$69).

and in the NATUG Progress Reports series

- ▷ NATUG–1, 5–6 April 1989, Salt Lake City, Utah, ed. G. S. Stiles (pp. 166, fl.120/ $60).
- ▷ NATUG–2, 18-19 October 1989, Durham, North Carolina, ed. John A. Board, Jr. (pp. 462, fl.230/$115).
- ▷ NATUG–3, 26–27 April 1990, Santa Clara, California, ed. Alan Wagner (pp. x+352, fl.190/$ 95).

IOS also publish the most recent proceedings of others of the user groups,

- ▷ Proceedings of the 3rd transputer/occam international conference, Tokyo 1990, ed. Tosiyasu L. Kunii and David May (pp. x+308, fl.170/$89).
- ▷ ATOUG–3, Proceedings of the 3rd Australian transputer and occam user group conference, Melbourne, 1990, ed. T. Bossomaier, T. Hintz and J. Hulskamp, (approximately 200 pages, fl.120/$60).

and the proceedings of the international conferences on the applications of transputers organised under the auspices of the UK SERC/DTI 'Transputer Initiative'.

They are available from bookshops or direct from the publisher, either individually or on continuation orders.

IOS
Van Diemenstraat 94
1013 CN, Amsterdam
The Netherlands
Fax: +31 20 22 60 55

or in the USA and Canada
IOS
Postal Drawer 10558
Burke, VA 22009-0558
United States of America
Fax: +1 703 250 47 05

or in Japan
IOS Japan Department
Highway Development Co. Ltd
1st Golden Building, 8-2-9 Ginza
104 Tokyo – Chuoku
Japan
Fax: +81 35 72 86 72

SIG publications

The proceedings of last year summer's international conference organised by the Artificial Intelligence SIG have appeared as the first volume of the *Wiley Communicating Process Architecture Series*, series editor David May.

▷ *Parallel Processing and Artificial Intelligence*, ed. Mike Reeve and Steven Ericsson Zenith, (pp. xvi+291; £29·95); John Wiley & Sons, 1989.

NEWSLETTER BACK ISSUES

The first ten issues of the *OUG Newsletter* are now out of print. The principal information of continuing value in the early issues is the bibliography of articles related to occam and the transputer. I have all this information in a text file on VAX, and when I have worked out the easiest way to turn this into an acceptable printed document I will be producing a bibliography booklet which will be sent to all members.

Additions to the bibliography and to the list of members will no longer appear in the *Newsletter* as we now have plenty of material without them. Supplements to or re-issues of both these documents will now be produced on an ad hoc basis, but we hope at least once a year. These will be mailed to everyone on the occam user group membership list. *Michael Poole, INMOS*

FORTHCOMING

TROLLIUS WORKSHOP

Cornell Theory Center, Cornell University, Ithaca, New York
24–26 July 1990

The Advanced Computing Research Institute (ACRI) of the Cornell Theory Center will be offering a three-day workshop on the Trollius programming environment for distributed-memory multicomputers.

Attendees will learn how to program and monitor parallel applications under Trollius Release 2.0. In addition, sessions will be offered on the design of the Trollius operating system, on use of special programming tools, and on porting Trollius. Guest speakers from industry will discuss related development projects.

Attendees will gain hands-on experience with the ACRI's transputer-based multicomputers, and will be given an opportunity to work on individual application programs. Attendees who are committed to using Trollius for existing projects are invited to remain through July 27, for in-depth consulting.

Attendees should have previous experience with UNIX systems, and in programming with C or FORTRAN. Previous experience in parallel programming is not required.

The workshop will begin at 9:00 am on Tuesday, July 24 and end by 4:30 pm on Thursday, July 26. It will take place in the Theory Center Training Facility, Engineering/Theory Center Building, Cornell University, Ithaca, NY.

Accommodation

The following accommodations are within walking distance of the Theory Center Training Facility and many restaurants and shops. Attendees are responsible for their own reservations and meals. Paid parking is available three blocks from the Training Facility. Commercial airport limousine service is also available.

Collegetown Motor Lodge Half a mile from Theory Center: 312 College Avenue, Ithaca, NY 14850; 607-273-3542, \$59–\$61 per night. Blocks of rooms have been reserved for this workshop. Please mention this when making your reservation.

The Peregrine House Bed and breakfast, half a mile from Theory Center: 140 College Avenue, Ithaca, NY 14850; 607-272-0919, \$62–\$72 per night including breakfast.

Registration

Registration is limited to thirty people. The registration fee is \$75.00 for academic attendees, and \$150.00 for corporate attendees. This includes course materials and refreshments at breaks. Applications and payment must be received by June 12, 1990. Applicants should receive notification of the final selection by June 22.

To apply for this workshop, please contact the Conference Coordinator: e-mail registrations may be sent to `donna@tcgould.tn.cornell.edu`, and surface mail registrations and payment may be sent to

Donna Smith Tel: +1 607 255-8686
Conference Coordinator
Theory Center
Cornell University
265 Olin Hall
Ithaca
New York, 14853-5201
United States of America

THIRTEENTH OCCAM USER GROUP TECHNICAL MEETING

University of York, England
18–20 September 1990, York

The Occam User Group invites all those interested in the Programming and Application fo Transputer based Architectures to attend its thirteenth technical meeting which will take place at the University of York from 18th to 20th September 1990. The meeting will include invited speakers, submitted papers, exhibitions, a panel session with key speakers, and meetings of the Special Interest Groups.

Full details of the meeting appear in separate publicity, or can be obtained from the meeting organiser:

Dr Hussein Zedan
Department of Computer Science
University of York
York YO1 5DD
England

Tel: +44 904 432744
Fax: +44 904 432767
zedan@uk.ac.york.minster

EDINBURGH PARALLEL COMPUTING CENTRE FIRST ANNUAL SEMINAR

James Clerk Maxwell Building, The King's Buildings, University of Edinburgh
Monday 24th September 1990

The Edinburgh Parallel Computing Centre, inaugurated in January 1990, is an interdisciplinary focus for parallel computing projects involving research groups within the University of Edinburgh, with industrial partners and academic users throughout Europe. It incorporates research, service and commercial divisions, as well as participants from University departments, and affiliates from industry. Its function is to enable the exploitation of parallelism in all kinds of problem-solving by creating and making accessible expertise, tools and facilities. It supports applications work through research into the fundamentals of parallelism, development of support tools, parallel supercomputing services, consultancy, training and user support, and industrial liaison.

The Centre's principal computing resources include a 400-transputer Meiko Computing Surface, providing nationally networked multi-user access, a 64-processor Parsytec system, and a 4096-processor AMT Distributed Array Processor. It is also responsible for service on a second DAP and on a new Computing Surface of 64 i860 nodes, targeted on 'grand challenge' problems of science and engineering. The Centre also coordinates activities with other parallel hardware in participating departments, such as DEC Fireflies, a DEC VAX 6340 symmetrical multiprocessor, and a Sequent Symmetry.

The Centre is supported by industry (including a major contribution from Meiko Limited), the University of Edinburgh, Lothian Regional Council, and by the Department of Trade and Industry, with major recurrent funding from the Science and Engineering Research Council and from the Computer Board.

The Centre's first annual seminar provides an overview of its activities, the hardware configurations and the software environment, and presents a selection of research projects in science, engineering and artificial intelligence from the expanding community of some two hundred active users. It also offers demonstrations of the local computing resources, and of software developed on them. The Parallel Computing Centre subsumes the Edinburgh Concurrent Supercomputer Project, and the seminar continues the series begun by the Project's Annual Seminars. It is open to anyone with interests in the development and application of parallel computing.

The seminar begins with the formal opening of the Edinburgh Parallel Computing Centre by Professor E. W. J. Mitchell, Chairman of the Science and Engineering Research Council. The keynote speech will be given by Daniel Hillis, founder of

Thinking Machines Corporation.

Programme

08.30 Registration and coffee

09.30 Opening of the Edinburgh Parallel Computing Centre: Prof E. W. J. Mitchell FRS, Chairman of SERC; Sir David Smith FRS, Principal, University of Edinburgh

09.40 Edinburgh Parallel Computing Centre status: Overview: D. J. Wallace; Service and utilisation: M. W. Brown; Industrial affiliation: D. B. Mercer

10.30 Coffee

11.00 Keynote speaker – W. D. Hillis (Thinking Machines Corporation)

11.50 S. F. Reddaway (Active Memory Technology): Parallel data transforms

12.10 Computing surface software environments

12.30 Lunch and demonstrations

13.50 to 17.00 in three parallel streams:

- ▷ Applications – talks will include:
  - ○ Molecular graphics
  - ○ Theorem-proving using MISD parallelism
  - ○ Simulating a shared-memory multiprocessor architecture
  - ○ QCD on i860s; a UK Grand Challenge
  - ○ Neural nets for oil-well faces determination
  - ○ Database searching on the DAP
  - ○ Parallel implementations of the adaptive Hough transform
  - ○ Mapping commercial CFD codes onto Transputer systems
- ▷ Tools and methods – talks will include:
  - ○ Linda/Prolog
  - ○ Simulation tools for parallel computers
  - ○ Performance programming for i860s
  - ○ Cellular automata on Transputers: the CAPE environment
  - ○ Constructive Solid Geometry
  - ○ Genetic algorithms for process planning
  - ○ 3-d surface tracking
  - ○ The heteroarchitecture project: SIMD and MIMD together
- ▷ Demonstrations

Presentations will be twenty-minute talks. There will be eight in each stream, with a break for tea and coffee from 15.10 to 15.40.

17.00 Close

Venue

The seminar will be held in the James Clerk Maxwell Building at the King's Buildings, the University's 'science campus' in South Edinburgh. This is some ten miles from Edinburgh Airport, and two and a half miles South of the city centre.

Fees

The seminar fee is £55. This includes admission, lunch, refreshments, and a Directory describing system, utilities and application developments within the EPCC and its user community. There is no charge for industrial affiliates or registered academic users of the Centre's services.

Fees must be paid in Sterling, by cheque or banker's draft or credit card (Access/ Eurocard/Mastercard/Visa can be accepted). Fees are NOT refundable except in the event of cancellation of the Seminar.

Accommodation

Delegates can book accommodation for any nights from 20th to 24th September. This will be provided in the Pollock Halls, about one mile North of the King's Buildings and a mile South of the city centre. We are providing accommodation for several days in advance of the meeting for the convenience of those who are also attending the 13th Technical Meeting of the Occam User Group, which takes place in York shortly before the Seminar, on Tuesday–Thursday 18th–20th September.

Industrial Affiliation

An Industrial Affiliation Scheme exists to foster collaboration with industry and to promote awareness and exploitation of concurrent computing. The annual seminar forms part of a two day meeting organised for Industrial Affiliates. Members and Partners should complete and return the separate form which is being sent to them.

This seminar is sponsored by:

▹ DTI/SERC Transputer Initiative

and co-sponsored by:

▹ The British Computer Society (Parallel Processing Specialist Group)

▹ The Institute of Physics (Computational Physics Group)

▹ The Scottish Development Agency

Contact for more information:

CEP Consultants Ltd
26–28 Albany Street
Edinburgh EH1 3QH
Great Britain

phone: +44 31 557 2478
fax: +44 31 557 5749

SYMBOLS VERSUS NEURONS?

Joint IEE and OUG Colloquium
IEE, Savoy Place, London
1st October 1990

Advance Announcement

This one-day meeting is being organised by the Institution of Electrical Engineers (Professional Group Committee C4 *Artificial Intelligence*) in collaboration with the

Occam User Group (Artificial Intelligence Special Interest Group). It will take place at the London headquarters of the IEE (Savoy Place, London WC2R 0BL) on 1st October 1990.

This colloquium will be the second International Conference of the OUG Artificial Intelligence SIG.

Aims of the Colloquium

In recent years, transputer-based parallel computers have gained in significance as a platform for the development of AI applications and tools. Sub-symbolic or neo-connectionist approaches are occupying a growing position at the side of classical symbolic approaches. This colloquium will highlight this development with a comparison between symbolic and connectionist approaches and their implementations. In posing the question 'Symbols versus Neurons?' a forum will be provided for directly tackling the central conflict of the AI debate today.

Conference Programme

Keynote speakers

▷ Professor Tom Addis (University of Reading, UK), *Knowledge and the Structure of Machines*

▷ Pau Bofill (Barcelona Polytechnic, Spain) and José del Millan (EC, Ispra, Italy), *A Systolic Algorithm for Back Propagation: Mapping onto a Transputer Network*

Invited speakers

▷ Professor Kimmo Kaski (Oxford University, UK), *Simulating Neural Networks in Distributed Environments*

▷ Professor E. von Goldammer (Universität Lübeck, West Germany), *Neural Nets – Applications in Medicine*

▷ Professor André Bakkers (Twente University, Enschede, The Netherlands), *Applications of Neural Control*

▷ Dr Jean Sallantin (CRIM, France), *Artificial Intelligence for Genomic Interpretation*

▷ Dr Lyubomir Stoychev (IMS, Sofia, Bulgaria), *Relational and Differential Logic for Knowledge Processing*

▷ Dr Terence C. Fogarty (Bristol Polytechnic, UK), *Using the Genetic Algorithm to Adapt Intelligent Systems*

▷ Reem Bahgat (Imperial College, London, UK), *Symbolic Constraint-Based Reasoning in PANDORA*

▷ Joachim Stender (Brainware GmbH, Berlin, West Germany), *Machine Learning Applications on Transputers*

▷ Zoltan Schreter (Universität Zürich, Switzerland), *Connectionism – A Link between Psychology and Neuroscience?*

▷ Steffen Schulze-Kremer (Freie Universität, Berlin, West Germany), *Inductive Protein Structure Analysis using Transputers*

Further Information

The conference programme has been planned by Joachim Stender, chairperson of the OUG AI SIG. Further information on the content of this programme should be addressed to:

Joachim Stender/Eva Hillebrand	Tel: +49 30 463 30 58
Brainware GmbH	Fax: +49 30 469 46 49
Gustav-Meyer-Allee 25	
D–1000 Berlin 65	
West Germany	

or to:

Professor T. Addis	Tel: +44 734 875123
Department of Computer Science	Fax: +44 734 751994
University of Reading	
P O Box 220	
Whiteknights	
Reading	
Berkshire RG6 2AX	
United Kingdom	

Registration Details

This meeting will be advertised by the IEE in the normal manner for its colloquia series. All bookings will be handled by the IEE on their registration forms. (These are not yet available.) At the moment, it is anticipated that the registration fee will be the normal one (i.e. £26·50 for IEE members and £43·50 for non-members, pounds sterling). OUG members may also qualify for the reduced fee.

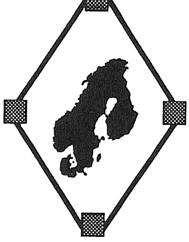
In the meantime, delegates wishing to attend should send their names and addresses to:

Professor P. H. Welch	Tel: +44 227 764000
Computing Laboratory	Fax: +44 227 762811
University of Kent	phw@uk.ac.ukc
Canterbury	
Kent CT2 7NF	
United Kingdom	

FIRST NORDIC TRANSPUTER SEMINAR

Provisionally located onboard a Viking Line ferry, between Sweden and Finland

5–7 October 1990



Call for speakers and abstracts

The aim of this seminar is to provide a forum for users of transputer-related technology in the Nordic countries. Researchers and application developers from academia, industry and government are invited to present results and experiences from their work.

The seminar

The provisional site for the seminar is a ferry between Stockholm and Helsinki. All standard conference facilities like slide- and overhead-projectors will be available. Themes for the seminar may include, but are not limited to the following:

- ▹ real-time systems
- ▹ super-computing
- ▹ artificial intelligence
- ▹ modelling and simulation with occam
- ▹ data acquisition
- ▹ future transputer and occam projects
- ▹ HW/SW design methods
- ▹ industrial applications

The seminar aims at giving a representative survey of current transputer activities in the Nordic countries. There should also be excellent opportunities for informal contacts with fellow transputer users. Workshops are being planned and we hope to involve all seminar attendees. We welcome suggestions on particular themes for workshops. We also welcome proposals from any person or group that wishes to run such a workshop.

Commercial companies will be invited to have exhibitions and demonstrations at the seminar.

Sponsors

- ▹ TH's Elektronik AB, Sweden
- ▹ Peder Pedersen AS, Denmark
- ▹ Tahonic AS, Norway
- ▹ ITT Disti, Finland

Programme committee

- ▹ Arne Sommerfelt, Senter for Industriforskning, Oslo, Norway
- ▹ Bjørn Rudberg, Forsvarets forskningsinstitutt, Norway
- ▹ Kari Leppälä, Technical Research Centre, Oulu, Finland
- ▹ Ralph Back, Åbo Akademi, Finland
- ▹ Jan Bardino, R.C. International, Århus, Denmark
- ▹ Jesper Nørly, Technical University of Denmark, Lyngby, Denmark
- ▹ Martin Tørngren, Royal Institute of Technology, Stockholm, Sweden
- ▹ Lars Estreen, Royal Institute of Technology, Stockholm, Sweden
- ▹ Niklas Ljung, TH's Elektronik AB, Sweden

Invited speakers

Representatives from Inmos will hold talks on the following topics:

- ▹ Future Inmos products, H1 the next generation transputer, the future of occam.

▷ Current status of the transputer, what market niche does the transputer hold in competition with other processors like Intel i860, Harris RTX 2000, Motorola 88000, AMD29000, i960, etc.

We are currently contacting several other potential speakers from central departments of transputer related activities.

Contributors

Speakers are invited to submit an extended abstract of 2–4 pages. Accepted abstracts will be collected in a seminar proceedings report, which will be distributed at the seminar.

The abstracts should be submitted in A4 format and ready for photo-copying. Please avoid colour photographs or other illustrations that are not well rendered by photo-copiers. Names, titles and institution of all authors should be on the first page of the abstract. If there are several co-authors the speaker at the seminar should be indicated.

In addition to the abstract we would like to have the full mailing address, telephone number, fax number and e-mail address (if available) of the author(s) to be notified. The abstracts should be mailed to the committee member closest to you in the list below:

Kari Leppälä
Technical Research Centre of Finland
Computer Technology Laboratory
Box 201 (Kaitoväylä 1)
SF-90571 Oulu
Finland

Bjørn Rudberg
Forsvarets forskningsinstitutt
Box 25
N-2007 Kjeller
Norway

Jesper Nørly
Technical University of Denmark
Building 421 - IK
DK-2800 Lyngby
Denmark

Martin Tørngren
DaMek/Machine Elements
Royal Institute of Technology
10044 Stockholm
Sweden

Deadlines

15 August 1990: Deadline for submission of extended abstract.
1 September 1990: Accepted speakers will be notified before this date.
1 September 1990: Deadline for seminar enrolment and payment of fee.
14 September 1990: Final seminar program is distributed.
5 October 1990: Ferry leaves Stockholm, returning on 7th.

Fees and accommodation

The seminar registration fee will be around 1000–1500 Swedish Kronor, the exact amount will be published later. The fee will include accommodation and meals on the ferry. Requests for further information and booking details regarding the seminar should be directed to:

Niklas Ljung	Tel: +46-8-362970
TH's Electronik AB	Fax: +46-8-7613065
Box 3027	
16303 Spånga	
Sweden	

NORTH AMERICAN TRANSPUTER USERS GROUP FALL MEETING

Cornell Theory Center, Cornell University, Ithaca, New York
10–11 October 1990

Call for papers

The 1990 Fall meeting of the North American Transputer Users Group will be held in Ithaca, New York, on October 11 and 12. Contributions are being solicited in the areas of hardware, software, and applications. Presentations will be allotted 20 minutes each, with ample additional time for questions; this format will allow for approximately 24 talks. Additional papers may be presented at poster sessions if there is sufficient interest. All accepted papers will appear in a published proceedings.

Abstracts of 1000–1500 words must be received by 15 June 1990. Notices of acceptance will be sent out by 1 July, and final drafts of accepted papers will be due by 1 August. Authors should indicate on the abstract whether they would be willing to present the paper in a poster session.

Abstracts (electronic mail submissions of the abstracts are preferred) should be submitted to:

David L. Fielding	+1 (607) 255-8686
Cornell Theory Center	fielding@tcgould.tn.cornell.edu
Cornell University	
265 Olin Hall	
Ithaca, NY 14853	
United States of America	

TRANSPUTING 1991

First world conference of national transputer and occam user groups
Santa Clara, California
22–26 April 1991

The first world conference of all Transputer and/or Occam User Groups will take place at:

TRANSPUTING 1991	
Place:	Santa Clara, California at the Sunnyvale Hilton
Dates:	week commencing 22nd April 1991

Call for Papers

The goals of this conference are:

▹ to present 'state-of-the-art' research on all aspects of parallel computing based upon communicating process architectures;

▹ to demonstrate 'state-of-the-art' products and applications from as wide a range of fields as possible;

▹ to progress the establishment of international software and hardware standards for parallel computing systems;

▹ to provide a forum for the free exchange of ideas, criticism and information from a world audience gathered from Industry, Commerce and Academia;

▹ to establish and encourage an understanding of the new software and hardware technologies enabled by the transputer;

▹ to promote an awareness of how these technologies may be applied and what their advantages are.

The conference themes will *include*: education and training issues, formal methods and security, performance and scalability, porting existing systems, parallelisation paradigms, tools, programming languages, support environments, standards and applications.

Applications *include*: embedded real-time control systems, workstations, supercomputing, consumer products, artificial intelligence, databases, modelling, design, data gathering and the testing of scientific or mathematical theories.

The conference programme will contain invited papers from established international authorities in these fields together with submitted papers. The international programme committee, presided over by Professor P. H. Welch (OUG Chair), is now soliciting papers on all areas described above. All papers will be fully refereed. Only papers of high excellence will be accepted. The proceedings of this conference will be published internationally and copies will be given to delegates when they register at the start of the meeting.

Programme committee

The members of the programme committee will be invited experts from Industry and Academia together with existing committee members from the joint organising user-groups based in Australia, France, Germany, India, Japan, Latin America, New Zealand, North America, Sweden, and the United Kingdom.

The organising load is being spread around the world, to ensure that all points of view and expertise are properly represented and to obtain the highest standards of excellence.

Instructions to authors

Seven copies of submitted papers (not exceeding 16 pages – single-spaced, A4 or US 'letter') must reach one of the Program Committee members nominated below before *1st October 1990.* Authors will be notified of acceptance by *15th November 1990.* Final camera-ready copy will be required by *6th January 1991.*

A submitted paper should be a draft version of the final camera-ready copy. It should contain most of the information, qualitative and quantitative, that will appear in the final paper, i.e. it should not be just an extended abstract. Please submit your paper to the committee member closest to you in this list:

Professor P. H. Welch
Chair, OUG
Computing Laboratory
The University
Canterbury
Kent CT2 7NF
United Kingdom
Tel: +44 227 764000
Fax: +44 227 762811

Dr D. Stiles
Chair, NATUG
Dept of Electrical Engineering
Utah State University
Logan UT, 843222-4120
United States of America
Tel: +1 801-750 2806
Fax: +1 810 750 2992

Professor T. L. Kunii
Chair, OUG Japan
Dept of Information Science
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo
Japan
Tel: +81 3 816 1783
Fax: +81 3 818 4607

Dr K. S. Venkatesh
Centre for the Development
of Advanced Computing
2/1 Brunton Road
Bangalore – 560 025
India
Tel: +91 812 575781
Fax: +91 812 563183

Dr J. Hulskamp
Chair, AOTUG
Department of Communication
and Electrical Engineering
Royal Melbourne Institute
of Technology
GPO Box 2476V
Melbourne 3001
Australia
Tel: +61 3 660 2453
Fax: +61 3 662 1060

Dr R. Lins
Chair, OUG: Latin America
Department de Informatica
Universidade Federal
de Pernambuco
P O Box 7851
50739–Recife–PE
Brazil
Tel: +55 812 713052
Fax: +55 813 264880
(The British Council)

Conference structure

```
PAR
  ... exhibition / demonstrations (Monday .. Friday)
  SEQ
    ... seminars / workshop (Monday)
    ... conference papers (Tuesday .. Thursday)
    ... seminars / workshop (Friday)
```

Call for tutorials and workshops

On the Monday of Transputing 1991, we shall be holding some tutorials on the fundamental principles underlying *transputer* technologies and various design paradigms for exploiting them. On the Friday, we shall be running several workshops that will focus directly on a range of specialist themes (e.g. real-time issues, formal methods, AI, super-computing, ...).

We welcome suggestions from the community for particular themes which should be chosen for these tutorials and workshops. In particular, we welcome proposals from any group that wishes to run such a tutorial or workshop. A submission should give details of the proposed programme, numbers of people to attend (minimum and maximum) and how much equipment (if any) is needed for support. Please submit your suggestions and/or proposals to one of the Committee listed above by *1st October 1990.*

Further information

Companies and other organisation wishing to be associated with Transputing 1991 should contact either Peter Welch or Dyke Stiles at the addresses above.

THE 4TH TRANSPUTER/OCCAM INTERNATIONAL CONFERENCE

Tokyo, Japan
21–24 May 1991

Following the successful and timely international third conference, OUG Japan is pleased to announce the *Fourth Transputer/Occam International Conference* to be held on 21st–24th May 1991 in Tokyo. This year we set 'Concurrency: from formalism to implementation' as the main theme of the conference. The topics include (but are not restricted to):

- ▷ formalism, specification and design;
- ▷ concurrent system development;
- ▷ application cases;
- ▷ practice and experiences;
- ▷ hardware and software;
- ▷ architecture.

This series of conferences has been a place to find a wide variety of Transputer based parallel processing systems through original refereed papers, invited talks, tutorials and commercial exhibits. The official language of the conference is English. Simultaneous translation from English to Japanese is provided whenever necessary. The registration fee is ¥40 000 and this includes:

- ▷ a copy of the conference proceedings;
- ▷ admission to all technical sessions, tutorials and exhibits;
- ▷ admission to all lunches;
- ▷ admission to the final party.

Call for papers

Full papers in English are solicited and should contain previously unpublished original high-quality results.

Please submit five copies of the paper to:

Prof. Dr Tosiyasu L. Kunii
Program Chairman
The 4th Transputer/Occam
International Conference
Department of Information Science
Faculty of Science
The University of Tokyo
7–3–1 Hongo
Bunkyo-ku, Tokyo 113
Japan

Phone: +81 3-812-211 x 4116
Fax: +81 3-818-4607
kunii@is.s.u-tokyo.ac.jp

The submitted paper should be typed, double spaced, about 10–20 pages in length with an abstract of 100–200 words and a maximum of 10 keywords. All papers will be peer reviewed and will be assessed with regard to their quality and relevance. The proceedings will be published for distribution at the conference and also for later dissemination. Submitted papers must be received by the program chairman by 1st December 1990. Acceptance or rejection of the papers will be notified by 15th January 1991. Final camera-ready papers must be received by the program chairman before 10th March 1991.

For tutorial proposals and any suggestions contact:

Dr David May
INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
United Kingdom

Tel: +44 454 616616
Fax: +44 454 617910

For local arrangements contact:

Mr Kazuto Matsui, Secretary
Occam User Group, Japan
SGS-Thomson Microelectronics KK
INMOS Business Centre
Nisseki Takanawa Building 4th Floor
18–10 Takanawa 2-chome
Minato-ku, Tokyo 108
Japan

Tel: +81 3-280-4125
Fax: +81 3-280-4131

REPORTS

NORTH AMERICAN TRANSPUTER USERS GROUP FALL MEETING

Duke University, Durham North Carolina
18–19 October, 1989

John A. Board, Duke University

The second 'official' meeting of the North American transputer users group was held on 18–19 October 1989 in Durham, North Carolina; Duke University served as academic sponsor of the event. Approximately 120 researchers from the USA, Canada, and five other countries were joined by about a dozen vendors of transputer-related products at this most successful meeting.

A full technical program over the two days included 26 delivered papers and several more poster contributions covering a very broad range of hardware, software, and applications topics. Two invited talks described major transputer projects under way in the United States. T. Bowen *et al.* from Bellcore are using transputers to prototype large high performance distributed databases that are essential in providing telephone services. D. G. Shea from IBM Yorktown Heights described his group's experience in designing and now using VICTOR, a family of transputer research workstations with from 16 to 256 nodes.

A session on performance evaluation tools provided evidence of the increasing hardware and especially software lengths people are willing to go to for more detailed information about the performance of their algorithms on transputer networks. A hardware and applications session described data acquisition and control applications of the transputer, a transputer-based computer input device for disabled persons, and the use of transputers for dataflow applications and in hypercube architectures.

Communications and network topologies were the subject of a session in which strategies for run-time reconfiguration and deadlock-free packet switching were discussed. Systolic computing on transputer arrays also received attention.

The second day of the meeting was dominated by discussions of applications and software environments. A session on physical and mathematical applications described work in parallel neural network simulation, parallel simulated annealing, and fast parallel multiplication. There is increasing interest in operating systems and message routing/task farming kernels for message passing architectures; papers on the current status of Linda and ALPS (from Ohio State) along with a paper describing a graphical design environment for transputer systems development filled a session on operating systems and environments. The issue of operating systems for transputer networks is of such importance to users here that a special session on this subject was planned for NATUG 3, to which all developers of such systems kown to us were invited.

The final session on software and algorithms described several efforts in bringing logic (AND/OR process models) and object-oriented (C_NET and C++) programming to transputer systems. There were also discussions of implementing bin-packing algorithms and wavefront algorithms on transputer networks.

All attendees were interested, as always, in comments from various SGS Thomson/INMOS representatives about present and future transupter products. I think my personal feelings are proably typical of those of many in attendance; I have been worried about stagnation in the transputer family especially in light of the rash of 'killer micros' – the various high performance RISC chips – announced in the last year, but I am also encouraged by the renewed vigour Thomson seems to be bringing the the transputer effort.

John Board jab@dukee.egr.duke.edu
General and Program Chair, NATUG 2
Dept. of Electrical Engineering
Duke University
Durham NC 27706
United States of America

LETTER FROM BULGARIA

Hugh Glaser and Chris Jesshope, University of Southampton

In March we attended the Workshop on Parallel Distributed Processing in Sofia, Bulgaria, organised by the Centre for Informatics and Computer Technology on behalf of the Bulgarian Academy of Sciences.

The Academy and Centre have had long-standing links with UK academic institutions, notably the Dataflow and Parsifal projects at Manchester, as well as Functional programming at Imperial and Electronics at Southampton. A number of these contacts were present, as well as academics from the City University and universities in Belgium, France, Greece, East and West Germany and the USSR.

Any of the participants at the workshop would expect a report to begin by noting that the social programme was very enjoyable. As most of us arrived early there was time to view the sites of Bulgaria. We visited the famous Rila monastery, a remarkable Escher-like construction, and took a stroll at the top of a (almost) snow-covered mountain, as well as seeing some of the beautiful countryside. One aspect of Bulgaria that Western visitors always find difficult to reconcile is the staggering difference between city and country life; it appears like a hundred year discontinuity in time. Continuing with the social aspects, the food was always acceptable, and often quite excellent. The organisation of both the social and technical programmes seemed to run effortlessly (although this was only an impression), and our hosts made strenuous efforts to ensure we were comfortable.

The workshop programme was split about half and half between Bulgarian presentations and foreign visitors. Since much of the non-Bulgarian work is familiar to UK researchers, it is perhaps more interesting to discuss the Bulgarian contributions.

Much has clearly happened since we attended the previous workshop one year ago. The political changes have probably not changed the technical directions very significantly, but they do now suffer from the familiar uncertainty about future funding levels as they await the decisions of the politicians, who in their turn await the decision of the people. The subjects of the presentations were quite different, however. Last year there was a considerable variety of subjects: as well as papers on networks and transputers, there were papers on functional and dataflow programming, and some excellent descriptions of their vector-processing supercomputer and high fidelity numerical analysis package; this year the emphasis was almost entirely on networks, transputers, and their use. The only variety was provided by the foreigners.

Apart from the social and technical programmes, we were invited to visit the institutes and one of their manufacturing plants. At the institute we were shown their supercomputer and their prototype multi-transputer system (48 transputer), which they expect soon to extend to a 72 transputer machine. This year we were shown around the Pravez personal computer plant, and it was rather intimidating to see the level of investment that has taken place, along with the capacity for expansion. It was also rather exciting to finally spot a British machine among all the US and West German hardware.

Towards the end of the workshop our hosts reported, with some excitement, that they had been asked to consider the possibility of holding next year's occam user group meeting in Bulgaria, perhaps on the Black Sea. Although neither of us have ever visited this area, it is well known to us through information provided by a Bulgarian research student who has been working in this department. The climate is Mediterranean and the sea is consistently warm, being 18–25°C during May to September. What is more, during one of the excellent meals in Sofia, one of our hosts entertained us with stories of how he had manipulated Bosses and Unions ruthlessly in order to acquire a yacht for his institute, for the recreation of the employees. Who knows; it may be OUG at sea! On the evidence of this visit it is clear that such a meeting would be both enjoyable and interesting.

Hugh Glaser — Tel: +44 703 593670
Department of Electronics and Computer Science — Fax: +44 703 593045
University of Southampton — Telex: 47661
Southampton SO9 5NH — hg@uk.ac.soton.ecs
United Kingdom

POSTCARD FROM EXETER

Tony Curtis, University of Kent

Even the most neutral of observers would have noticed a biblical theme to the Exeter OUG. A comment to the effect of 'I expect it will snow next' proved to be all too prophetic. I wonder where the plagues of locusts went? As is usual for Devon, the weather played its normal tricks, however we plucky delegates ploughed on regardless of the climatic variations.

The campus itself is rather on the hilly side, and has not completely succumbed to a proliferation of concrete at every available site. Thus there are still plenty of wooded and green areas on campus, which are nice to see.

On to the conference itself. It followed the usual three day format with papers, exhibitions from various companies and product announcements, and of course gastronomic excesses.

The papers were as varied as one might expect. Most aspects of computing were represented from real-time control of robots to graphical programming environments. There were some useful ideas about monitoring channels in one talk through to transformational methods for program generation in another. Something for everyone really. One speaker was unable to attend due to illness, but the day was saved by Peter Welch who nimbly stepped in and gave a talk about standards and safety-critical systems.

The remuneration for the presentations was a bottle of beer (which is apparently lethal and should not be drunk until the next century!) and a bottle of wine (presumably less damaging).

The lecture theatre was the largest one in the Statistics department, which was where all the talks and exhibitions were held. The lecture theatre itself showed distinct signs of having been in the middle of a botanical explosion with flowers strewn along the stage at the front. Fortunately no speakers became lost in the flowers.

The exhibitions were held along the corridor outside the lecture theatre. Both software and hardware were represented ably by companies such as Transtech, Division and CSA, who had an inexpensive transputer board on display. INMOS also gave a product announcement during which the availability of the source of an occam compiler was indicated. This should certainly open up the use of occam to other systems; occam needs to be seen as accessible if it is to gain acceptance outside of purely transputer-based systems (i.e. abstracted from the hardware aspects).

The culinary content of the conference was increased somewhat by the conference meal on the Tuesday evening. It was held in the long refectory in Devonshire House, which is the main Student Union building and is also where the lunches were held. An inpromptu speech from Derek Partridge, who is head of the computer science department at Exeter wound up the meal; for someone who was not expecting to have to give a speech he extemporised very well!

Slotted into all this high living were the Special Interest Groups (SIGs). Unable to clone myself, I only attended three.

The graphical tools SIG focused on the type of information that should or could be extracted for real-time monitoring. Other topics of discussion included windowing programming tools for parallel programming.

The operating systems SIG had become just 'environments'. This did not seem to stop people talking about operating systems. Naturally, the main ones discussed were those with which attendees had had experience. These included MEiKOS, Helios, and GENESIS amongst others. No one knew all that much in detail about Mach or Chorus, but the general consensus was that this seemed to be about the right way of doing things. The emergence of Amoeba in the near future was also mentioned. The SIG was also adjudged to be interested in programming tools for parallel processing. Specific mention was given to MEiKO's CS Tools package as a usable system.

The third SIG I attended was the AI one. This spent all its time talking about the organisation (or not) of a one-day conference or colloquium which will focus on the two main threads of AI research today, namely the neural versus the symbolic approach (see page 9).

As ever, the Roger Shepherd Memorial Joke Contest was held toward the end of the conference. There appeared to be a distinct dearth of jokes this time, with some efforts being scribbled down just beforehand. The prize went to a University of Kent concoction, this being the one about occam programmers and sharp culinary tools. On the whole though, one feels that this selection of jokes lacked a certain something. As for why it is a 'memorial' contest, who knows? Roger looked well enough, or does he know something we do not?

The big debating point was about the change of name from 'occam user group' to something which reflected the group's activities more closely. The 'Association for Transputing Machinery' seemed to be a favourite, though personally I would like to retain a reference to the software side in the name. As Occam's Razor doesn't say...

Nomini non sunt multiplicanda praeter necessitatem

Names should not be multiplied beyond necessity

so perhaps the list from which to choose should be kept small.

The conference was organised by the computer science department at Exeter. Special thanks are due to Stephen Turner for all the work he put into keeping us in order and editing the proceedings. Thanks are also in order for the secretarial and support staff who kept everything else going.

HOW MANY LINKS DOES IT TAKE TO SCREW IN A TRANSPUTER?

In celebration its patron's re-appearance at the occam user group meeting in Exeter, there was held a Roger Shepherd Resurrection Joke Contest. The man himself was prevailed upon to read out the contributions, subject to some real-time pre-censorship, and amongst the printable contributions were:

Q: How do you tell an Inmos salesman that you need an array of T400s?
A: Give him a *ring* or hit him with a *pipe*!

That got a very frosty reception, as did

Q: Why do you keep transputers in anti-static foam?
A: To stop them getting a shock from their community charge.

For non-UK readers 'community charge' is the official name of a relatively new and absolutely unpopular local tax in Britain. (Whoever heard of a popular tax?)

Q: Why do occam programmers eat with their fingers?
A1: Security forbids their using forks.

Me I liked that one, but then there was

A2: They don't use knives because common sense suggests they aren't allowed sharp objects.

and to growing audience sympathy

A3: Because they aren't used to using advanced tools.

Cutting. Very cutting.

Q: What have the evolution of the T400 and man in common?
A: The missing link(s).

Q: What's the difference between an ant scratching and an oug speaker?
A: You can hear an ant scratching.

Well, perhaps if you had been there...

Q: What will still have eighty-four legs?
A: The H1 design team if it meets Pistorio's schedule.

Best not to comment on that one perhaps.
Q: Why don't occam programmers use suitcases?
A: Because they fold all their clothes up and put them in their pockets.
Then there was a signed contribution from *The Lads* describing a number of familiar-sounding names:

Division 'The Pirhana brothers'
Transtech 'East London wide boys'
Meiko 'Arthur Daley Software'
Inmos 'Hammer and Axe boys'

If you don't understand that, all the better for all concerned. (The editor of the newsletter, the publishers, and my maiden aunt all wish to make clear that any resemblance of the purely fictional characters in the preceding joke to companies or persons alive or deceased is purely for the purposes of ridicule, and has nothing to do with us, squire; you want The Lads.) It seems a bit hard on Meiko who seemed not to be represented at the exhibition this time, despite having paid for a stand.

Your correspondent found himself unexpectedly pressed into service as clap-o-meter and on the basis of audience response awarded the first choice of bottle to Colin Willcock (have regulars readers heard this name before?) for the advanced tools, and the second bottle to John Fisher for the stridulating hymenopterous emmet. (Why is my spelling checker complaining?) It was not clear which winner came off best in their choice between red plonk and a very small bottle of a very potent brew that needed to be preserved into the next century before being drunk. *gj*

NORTH AMERICAN TRANSPUTER USERS GROUP SPRING MEETING

Sunnyvale, California
26–27 April 1990

Alan Wagner, University of British Columbia

The third NATUG meeting was held April 26 and 27th in the center of the Silicon Valley. About 120 people attended the meeting, approximately 10% of the participants were from outside of North America. In total there were over 30 presentations, 21 papers in the regular sessions, 6 papers in a special session on software systems and environments, and a further 4 papers in the poster session. There were 13 vendors present with a variety of hardware and software products. The proceedings have been published by IOS Press under the title *Transputer Research and Applications 3.*

As usual, the meeting attracted papers from throughout the world. About 40% of the presented papers were from outside North America. The international participation was most welcome and gave us the opportunity to learn more about what our overseas counterparts are doing. The papers broadly fell into two categories: applications, and software systems and environments. The first day was devoted to transputer-related algorithms and applications, several of which involved specially designed hardware or transputer configurations. The technical sessions were as follows:

- ▹ *TransAcq: real-time data acquisition and analysis system for optical spectroscopy,* Brad V. Duncan, Linda S. Powers and G. S. Stiles;

- ▷ *The development of a visual telephone for the deaf: using transputers for real-time image processing*, Scott Galuska;
- ▷ *Using the transputer in the design of high performance architectures dedicated to the implementation of OSI transport protocol*, Christophe Diot and Michel Ng. X. Dang;
- ▷ *TCP/IP Networking using transputers*, Roger M. A. Peel;
- ▷ *Revised simplex method on a network of T800 transputers*, J. Luo, F. Bruggeman and G. L. Reijns;
- ▷ *Transputer implementation of the EM algorithm for PET image reconstruction*, Fred U. Rosenberger, Gerald C. Johns, David G. Politte and Charles E. Molnar;
- ▷ *The fast multipole algorithm on transputer networks*, John A. Board, Jr. and James F. Leathrum, Jr.;
- ▷ *A parallel-processing subsystem for the generation of 3D cardiac images from CT*, Scott R. Cannon and Stephen J. Allan;
- ▷ *Molecular dynamics simulations on a systolic ring of transputers*, K. Boehncke, H. Heller, H. Grubmuller and K. Schulten;
- ▷ *Simulation of self-organizing neural nets: a comparison between a transputer ring and a connection machine CM-2*, K. Obermayer, H. Heller, H. Ritter and K. Schulten;
- ▷ *Parallelizing, using process-and-data-decomposition (PADD) approach on a multi-ring transputer network – an example*, Hamid R. Arabnia and Mary R. Robinson;
- ▷ *Hardware voting of transputers in real-time nMR fault-tolerant systems*, J. Standeven and M. J. Colley;
- ▷ *Transputer-based multi-robot simulation*, Hubertus Franke, D. Shea and L.C. Zai;
- ▷ *Partitioned, replicative neural networks for co-operative robot systems*, Martin J. Dudziak.

The last talk of the day was by Andy Rabagliati from Inmos on the H1, the next-generation transputer. He discussed mainly the memory organization of the H1.

The morning of the second day was devoted to the special session organized by Dyke Stiles on topology-independent programming systems. Each participant gave a presentation and the talks were followed by a panel discussion. The presentations included:

- ▷ *Deadlock-free parallel programming on transputer networks*, Prasad Vishnubhotla from Ohio State University,
- ▷ *The Express/Cubix system*, Adam Kolawa from Parasoft;
- ▷ *Helios*, Jason Cockroft from MIMD Systems;
- ▷ *System-level parallel programming based on Linda (a call to standards)*, Wm Leher;
- ▷ *Dynamic parallel programming*, Jan Graat from Parsec;
- ▷ *Muliptle language programming with STRAND*, John Florentin from Birkbeck College;
- ▷ *Process level parallelism in a UNIX environment*, Paul King from Real Time Systems;
- ▷ *The Trollius programming environment for multicomputers*, Dave Fielding *et al.*

It was clear from the presentations that transputer languages, systems and environments have matured considerably over the last year. These systems provide at one level or another a topology-independent approach to programming transputers. I hope that in the coming year as these systems become more widely available we will all have a better opportunity to evaluate them.

The afternoon of the second day was devoted to software systems and environments. The following papers were presented:

- ▹ *Multigraph for the transputer*, Ben Abbott, Csaba Biegl and Janos Sztipanovits;
- ▹ *Graphical visualization of distributed algorithms*, Oliver Vornberger and Klaus Zeppenfeld;
- ▹ *A visual programming system for the transputer*, M. Roberts and P. M. Samwell;
- ▹ *An efficient and flexible implementation of ALT*, S. W. Lau and F. C. M. Lau;
- ▹ *A dynamic distributed system using remote procedure calls in a message passing system*, Tom Hintz and Mark Phillips;
- ▹ *A virtual architecture for investigating dynamic load balancing on transputer networks*, I. A. Horton and S. J. Turner;
- ▹ *Mapping search graphs onto arbitrary transputer networks (or making PROLOG parallel)*, Douglas Eadline.

Posters were presented during the breaks on Thursday afternoon and Friday. These included:

- ▹ *On the performance of ALT in Occam*, K. M. Shea and F. C. M. Lau;
- ▹ *A transputer-based motion detection/tracking algorithm*, Andrew P. Bernat and James Rupel;
- ▹ *A transputer fault-tolerant processor*, Jorge L. Ortiz, Willie L. McCoy and Michael M. Thomas;
- ▹ *Applications of transputer interface to DSP vector processor*, Richard L. Tutwiler.

There were, for the first time, awards given for the best presentations. Ballots were distributed to all participants and tallied after the meeting. First place went to Wm Leher for his talk entitled 'System-level parallel programming based on Linda (a call to standards)' and second place went to Scott Cannon for his talk on a parallel-processing subsystem for the generation of 3D cardiac images.

NATUG 4 is scheduled for 11–12th October at Ithaca, New York (see page 14).

The first world transputer meeting is scheduled for April 1991 (see page 14) and will also be held in the Silicon Valley. Those of us who attended NATUG 3 can attest to the beauty of California at this time of year. Sunnyvale lived up to its name. Most of the social events were held pool-side in an informal and relaxed atmosphere. I would like to encourage our international colleagues to make the pilgrimage to Silicon Valley next year.

THE THIRD TRANSPUTER/OCCAM INTERNATIONAL CONFERENCE

Tokyo, Japan

17th–18th May 1990 *Kazuto Matsui, OUG Japan*

The third transputer/occam international conference was held in Tokyo. This time we invited key speakers from the UK, sponsored by the British Council and Inmos Ltd. The emphasis of this conference was to present recent progress, survey basic principles and architectural overviews through invited papers and tutorials including some parallel processing applications. A CSP short course was also provided by

Dr Jeff Sanders, Oxford University, in the tutorial session. The student session was an experiment to explore the potential of using transputers and occam to teach students, and doing research with them on parallel processing.

The CSP Short Course and student session were held at the Sanjo Kaikan Conference Hall on campus at the University of Tokyo on 15th and 16th May. The rest of the sessions and exhibitions were held at Seiryo Kaikan beside the Hibiya High School on 17th and 18th May. As the official language was English, we provided simultaneous translation from English into Japanese and vice versa. This enabled us to communicate well with each other throughout the conference.

CSP short course (Jeff Sanders)

Around a hundred people attended this short course, which was a first trial in Japan as far as I know. Dr Jeff Sanders prepared three texts for us ('An introduction to CSP', 'An incremental specification of the Sliding-Window Protocol', and his handout material) and taught us the main ideas of CSP. Initially there was a barrier to be overcome because most of the CSP notation was unfamiliar to us, but his superb teaching technique and his personality changed our attitude to learning CSP. All of the students, university teachers and industrial people were impressed by his teaching. Some people were were getting enthusiastic to learn more about CSP. Thank you, Dr Sanders! I sincerely recommend people in other countries that you should also take an opportunity to study CSP, otherwise you will not really understand what occam and the transputer are.

A modular, decentralised, transputer-based architecture for multi-sensor data fusion (Hugh Durrant-Whyte)

A fully decentralised architecture for data fusion problems was discussed. As this architecture takes the form of a network of sensor nodes, computation is performed locally and communication occurs between any nodes and this has many desirable properties.

Reasoning about distributed algorithms in CSP: application to remote sensing (Jeff Sanders)

CSP is a higher level language from which occam code can be obtained by routine down-coding but which is supported by a body of laws and a hierarchy of semantic models. That is why occam is used for the implementation of concurrent algorithms. Its simplicity and efficiency make it appealing for that purpose.

Student session

Parallel processing of quadtree images (Ryo Mukai)

To improve the speed of image processing, parallel processing of quadtrees was investigated. Parallelising the computation of convolutions could not be made to achieve a good efficiency because the algorithm and the structure of the network requires a great deal of communication. Eventually some people suggested a useful idea to solve this problem.

Divide and conquer in parallel processing (Shigeru Chiba)

'Divide and conquer' is a major technique in the solution of large problems, dividing them into smaller parts. Exploiting parallelism in this technique gains more efficiency. To implement this technique in parallel, both ring and tree topologies were tried and the ring topology was slightly faster than the tree.

Network and Topology (I)

Transputers and routers: components for concurrent machines (David May)

David May talked about why the router and latency-hiding method will be important when many transputers are connected. VLSI routers can provide routing between a large number of links, minimising network delays. Very fast routers with fewer links can be constructed using high-speed technology. Transputers and routers can be combined on VLSI chips to provide network nodes. This functionality will be supported by the next transputer.

Implementing functional languages on networks of transputers (Chris Howson)

A graph reducer for lazy functional languages using a network of transputers was reported. The main concern was correctly to transmit directed acyclic graphs from one transputer to another.

A general configurable multigrid implementation for the solution of three-dimensional elliptic equations on a transputer network (Osama El-Giar)

The implementation of a generally configurable version of the full multigrid method on arrays of transputers was made. The design of a parallel algorithm to allow for the easy distribution of computation for the case of Poisson's equation was presented.

Image Processing

A visual interface for a transputer network and its application to moving image analysis (Wiwat Wongwarawipat)

A VIT (Visual Image for Transputer) system was constructed to perform high-speed parallel image-processing on a transputer network. The major application of this system is to analyse moving-object image-data, projected onto an image plane, and to get the shape and displacement of the object at the same time.

Parallel 2D–FFT algorithm on practical multiprocessor systems (Kazuhito Itoh)

An algorithm for the direct computation of a two-dimensional fast Fourier transform was considered. Mesh, 2D torus and rectangular implementations were made. Although the interprocessor communication time increases, the number of multiplications can be reduced to three-quarters of that required for conventional indirect 2D–FFT.

Design and implementation of software-based real-time video codec using multi-transputer architecture (Masahiro Ichikawa)

A software based real-time video codec using multi-transputer architecture was implemented. The coding algorithm was based on the CCITT standard. By using several suitable parallel processing methods, a fully programmable real-time video codec was developed. Real-time operation will be achieved using QCIF at the rate of ten frames per second when one hundred transputers are used.

Network and topology (II)

TéNOR: a symbolic configurer for the SuperNode architecture (Jean Marc Adamo)

A symbolic configurer for the SuperNode machines was presented. TéNOR makes it possible to describe network configurations symbolically. TéNOR automatically checks whether the described topology meets the architectural constraints of the SuperNode machine and produces a configuration for the transputer network.

Implementing recursion on a double ring topology (J. L. Jacquemin)

Recursion was implemented on transputers by using a double-ring topology. This solution gives some major advantages. An efficiency reaching 90% can be expected.

Scientific applications

Scientific applications on transputer arrays: some experiments in MIMD parallelism (Tony Hey)

A survey of some problems inhibiting the use of parallel processing was presented, along with an overview of useful computational models and programming paradigms for parallel machines based on transputer arrays.

A design methodology for synthesising one-dimensional systolic algorithms (Youji Iihuni)

A method for synthesising 1–D systolic arrays from a serial algorithm having triple do-loops was proposed. Six systolic algorithms for the LD decomposition and two systolic algorithms for least-squares were synthesised.

A scientific application on a transputer system (Makoto Nishizaki)

A general scientific application system was implemented on transputers. A simulation program called CSSL (Continuous System Simulation Language) has been running only on sequential computers. A parallel implementation was developed using a PROLOG parser which generates occam source code. This could help many people who have to convert sequential codes into parallel ones.

Parallel algorithms (I)

Modelling queries in relational databases for parallel processing (J. L. Jacquemin)

Some research in optimising parallel queries in relational databases was presented. The feasibility of this modelling was explained, consisting of the following stages:

standardising query structures; building the query tree; transformation algorithm; and parallel processing.

Graph theory on a transputer array (James Allwright)

Finding the largest graph of a given valency and diameter is essentially the problem of network design expressed in the precise terms of graph theory. Here a heuristic algorithm to search for solutions to the problem was developed, based on Lin and Kernighan's algorithm for the travelling salesman problem.

Parallel algorithms (II)

A decentralised dynamic scheduling scheme for transputer networks (Satoshi Nishimura)

A new scheme for mapping dynamically created tasks onto a transputer network was proposed. The scheduler on each processor inspects the loads of the neighbouring processors and transfers excessive tasks to less loaded processors. It is now planned to implement the dynamic scheduling scheme on transputer networks.

Variations on the ALT implementation on the transputer (Dennis N. M. Ho)

This paper reported some of analysis and experiments with the occam ALT and PRI ALT constructs. It discussed the efficiency of the ALT realised on the transputer.

Progress towards an interactive solid modelling system on parallel computers (Peter Dew)

Research into parallel algorithms and systems is needed to support the computational geometry algorithms that arise in the automation of design and manufacture of solid parts and assemblies. The primary purpose of the system is to provide a realistic environment to evaluate the performance and scalability of parallel algorithms and software.

FAST ROUTING IN JAPAN

J. W. Sanders, Oxford University

The six-pm-Sunday direct flight from Heathrow to Tokyo is as kind a way of travelling to Japan as present airlines allow. The UK contingent – Peter Dew (Leeds), Hugh Durrant-Whyte (Oxford), David May (Inmos) and I – with only one smoker amongst four was split in inverse proportion to that of the JAL flight: our first warning of Japanese air space. In Tokyo we joined Tony Hey (Southampton, on leave at IBM Watson), the remaining invited speaker at the OUG conference starting in a day and a half.

But first a hundred or so of the participants had booked in for a $1\frac{1}{2}$ day CSP course. I took it, slowly, through an introductory overview, through the most important parts of Hoare's book, then through a case study on communications protocols. No matter how careful the English presentation, language is a problem in Japan; but handouts help, and the enterprising Kimio Maruyama had translated my *Introduction to CSP* into Japanese, for release with the video of the lectures.

Sixty more participants joined us for the third Japanese OUG meeting. In the first invited address Hugh Durrant-Whyte presented a decentralised design for Kalman filtering, and applied it to multi-sensor data fusion. In this lovely example of a distributed algorithm it is far from obvious that pairwise communications, achieved by relay through intermediate nodes, are done consistently. Proof of that fact would be an interesting exercise. (Durrant-Whyte had obviously confirmed it for some representative cases.)

In his address David May discussed the consequences of Valiant's observations on general-purpose parallel machines, introduced a router chip, and showed how it might be used effectively. One important comment that seemed to lack general appreciation: extra concurrency (allocation of excess processes to processors) helps to cover communication delays between processors. That is to be contrasted with the impulse to assign one process to each processor in order to gain maximum 'efficiency'.

Tony Hey surveyed MIMD parallelism and neatly reinforced May's talk by reporting experiments that measured execution time. He drew attention to the need for portability and generality of parallel machines and to the need for higher-level languages to drive them.

Peter Dew reported progress on a transputer-based solid-modelling system, called *Mistral-3*. The idea was to choose an architecture that permitted parallelism to be determined by the application. For that, care was taken in the choice of data structures: a hierarchically-ordered spatially-divided solid model was selected, and decentralised operations (like ray tracing) implemented in terms of it.

For further details, and for the other talks, the reader is referred to the proceedings (*Transputer/occam Japan 3*, edited by T. L. Kunii and D. May, IOS Press, 1990; see page 4 of this newsletter, also the preceding report on the meeting itself). However it is worth mentioning the high standard of two student papers. Both were given by undergraduates in the Kunii Laboratory at the University of Tokyo and both experimented with the use of tree-like networks of transputers. That evening we visitors were taken to dinner by Professor Kunii at a restaurant he has been frequenting for thirty years. I had the pleasure of discovering, with the aforementioned students, recursive definitions of graphs of small diameter; we found them to be more accessible than the usual pictures (like that of the Petersen graph).

After dinner, at about 9:30 pm, Kunii showed the visitors over his laboratory. His secretary was still there, working hard. On Friday evening after the conference I travelled to Sendai by *Shinkansen* (the bullet train) with Professor Nakamura. En route, at 9:55 pm, my companion's watch alarm sounded. He explained that on a normal working day that indicated he had 15 minutes to catch the last bus home. It seems that much of evening television in Japan is aimed at women; men stay in their offices until quite late. Indeed the day's *sumo* bouts are only covered at about 11:00 pm.

On Saturday most of the other visitors headed home from Tokyo. In Sendai I began phase two of my trip: visiting various labs selected by the British Council. I began with Nakamura's group, including Kobayashi and Horiguchi, in the Department of Mechanical Engineering at Tohoku University. Their primary interest is in general-purpose pipelines and, with experience of three such pipelines behind them, they prefer the transputer one—largely because of the ease of programming in occam. On Saturday morning about thirty students attended my seminar outlining how Z and CSP could be used to capture complementary aspects of system behaviour. I was proposing the use of mathematics for the description and development of systems that took for granted sequential techniques. I fear that my analogy, at one point, with loop invariants drew no response from the audience. Here I met for the first time in Japan the strong desire to reason about concurrent systems using some formalism more abstract than a programming language; yet the complete unfamiliarity with the methods for achieving that for sequential systems. It was to recur at almost every site.

That evening Mrs Nakamura produced an amazing feast of traditional dishes in spite of the fact that she had been teaching mathematics until early that afternoon. I gathered that the meal was unusually traditional even for them. After dinner we discussed the position of logic and set theory in Computation, aided by a textbook Nakamura had been working through at home. That, in an Engineer, shows real commitment to mathematical methods.

Sunday, a free day, I spent with family friends who had lived in Australia for three years 25 years ago. From the husband's example I learnt that it is not unusual for Japanese professors, after retirement, to be re-employed at the same or another department; he was building up a department in a new private university. From the wife's example I learnt that even in Japan women can be distinguished: in this case for writing and teaching *haiku.*

On Monday I visited the Noguchi lab (Applied Information Sciences, Tohoku University) which included Nunokawa and Togashi. Here I found that lecturers and graduate students were working on similar problems and with the same approach to those of people in Oxford. They were in the process of evaluating notations for describing development of parallel systems and, though not familiar with CSP, knew the basics of CCS; that made my task of explanation much easier. Ph.D. students were working on Multicast protocols and implementing concurrent Prolog using CCS.

That afternoon the *Shinkansen* took me rapidly back to Tokyo; next morning another bore me to Osaka. Passengers without reservations travel in pre-assigned carriages which are beset with a pall of cigarette smoke. To emulate, Mt. Fuji en route was capped with cloud. In my reserved seat I had, by this time, become addicted to the packed meals and cans of excellent orange juice served by *Japan Rail.*

That afternoon I visited Dr Deguchi, Electrical Engineering, Osaka University, (with Kondo and Araki) and we discussed models of CMOS design and CSP. Deguchi has translated Inmos's occam 2 book into Japanese and kindly presented me with a copy.

Next morning I was collected by Dr Katsuda and taken to Osaka Sangyo University where I spent the day. The visit had been arranged by Professor Yamamoto (whom I had met the previous week) with a precise purpose, and so it was structured more formally than other visits. His idea was to convey recent developments in parallel systems to a spectrum of his colleagues throughout the university. We began with a seminar presentation: myself on system specification and development; Professor Sugiura (Department of Economics) on a world-wide distributed project to simulate the world economy; Dr Hiratsuka (Department of Civil Engineering) on data flow in sewage treatment (cross my heart); and finally Yamamoto (Computing, The Junior College) summarising. It was a very impressive exercise in co-ordination.

Apart from those mentioned already, there were representatives from automotive design, video processing, and from a small Kyoto firm (about which more later). All recognised the need for concurrency in their area and all wished to find out about formalisms for describing it. Yamamoto had drawn them together and had been responsible for videoing my lectures on CSP in the previous week. His plan is to release a package for educational purposes, first within his university and then throughout Japan; I wish him luck.

The morning finished with a tour of the university's Computing Centre (the enthusiastic manager even handed out strands of optic fibre) followed by a marvellous lunch at a restaurant halfway up the hills beside the University. Unfortunately the view of the hills from Osaka is far more pleasant than the view of daytime Osaka from the hills! After lunch the group chose to have a tutorial-like session on the modelling of occam in CSP with emphasis on how to ensure the absence of deadlock and its application to their flexible-video project. That evening they took me to a marvellous restaurant where I was permitted to sprawl in un-Japanese style on the *tatami*. I can only agree with my tourist book: people in Osaka pride themselves on eating well.

Next morning I travelled the short distance to Kyoto by *Shinkansen*; of course I had a train lunch and orange juice for breakfast. Thus fortified, first stop was the group of Professor Katai (with Sawaragi) in the Department of Precision Mechanics at Kyoto University. They are interested in control theory and are experimenting with the use of occam and transputers for control systems and they had two nice examples: a qualitative model of a point's position on a curve; and a quantitative model for detecting fire. Again, the idea of a notation that could express their algorithms more abstractly than occam was welcomed with open arms. Particularly since those arms had had trouble with deadly embrace (as deadlock used to be called). Unfortunately on this occasion there was no time for me to work through a specific example.

After lunch I visited Professor Sakai in the Division of Applied Systems Science at Kyoto University. We had just an hour to discuss various approaches to systolic design. He and his colleagues are mainly interested in designs for least-squares. I was already familiar with the work because one of his students had presented a paper at the OUG meeting.

Before catching the *Shinkansen* back to Tokyo I visited Data Techno, a company with thirty employees which sells audio-processing equipment but is contemplating the move into more general transputer-based applications. Their research and development manager, Kojima, had been present the previous week in Tokyo and at the day's meeting in Osaka; I was curious to find out why such a small firm was so committed to finding out about a theoretical tool. Answer: it seemed self-evident to Kojima that they should package and sell transputers, that they would need to program them in occam, and that they would need to reason about their product. So simple.

As I left Kyoto for Tokyo and the airport I felt that if such companies are able to perform competitively with that degree of commitment to research and development, then we had better do our best to make the theory accessible to them. And I began to see why Japanese business has its present world standing; and was left to wonder what it would have been like to visit labs in some of the large companies.

SPECIAL INTEREST GROUPS

EDUCATION AND TRAINING SIG

Roger M. A. Peel, University of Surrey

The meeting started with a review of three hardware products which are becoming available at a low cost for educational institutions :

▷ Jon Kerridge (Sheffield University) has produced a T222-based Embedded Systems board. This incorporates 32Kbytes of RAM and 32Kbytes of EPROM, a multi-channel event controller, parallel ports, UARTs, analogue to digital converters, switches, lights, numeric keypad and a 7-segment display.

▷ Roger Peel (University of Surrey) has produced a 32-bit transputer-based single Eurocard, compatible with the lower connector of an ITEM rack. It is designed to support additional student project hardware, and therefore routes all the processor signals to an expansion connector, suitably protected against static and overloading. It also incorporates up to 8 Mbytes of DRAM, uniquely address-decoded to permit DMA enhancements, as well as four subsystem ports and programmable status lights.

▷ Most impressively, Computer Systems Architects (CSA) introduced an educational hardware and software package, produced in collaboration with INMOS USA. This is a PC-bus board supporting a 32-bit transputer, eight 4-bit wide memory sockets, headers to allow access to the memory and data bus, and Apple-compatible connectors for the INMOS Links. A version containing a T400 processor, no memory devices, restricted versions of Logical Systems C and the Occam Toolset, and 1000 pages of documentation is available at an introductory educational price of $236 in the USA. Repeat orders, without software or documentation, are available for just $150.

In response to a comment that there appeared to be few occam courses being advertised, Jon Kerridge reported that the Sheffield Transputer Support Centre was pushing occam hard. Its four-day occam courses introduced the other parallel languages for the transputer, too. Many of their attendees who had experimented with C, Pascal and Fortran had reverted to occam following their initial experiences.

Michael Goldsmith (Formal Systems (Europe) Ltd) announced that his new company was intending to run courses in Z, CSP and other specification techniques, as well as occam.

Various attendees agreed that they would like to see publications containing substantial examples of parallel occam, C and Fortran, especially general-purpose routing harnesses. The need to emphasise the process and procedural structures of an application at all levels of its abstraction was stressed.

Finally, more details of the Education and Training SIG Workshop were announced. On 3rd July 1990, the Sheffield Transputer Support Centre will be hosting a workshop to allow educators to program a parallel solution to a particular problem (a simulated controller for a photocopier), and then to discuss their solutions, as well as how best to teach the design principles utilised. The workshop will form the basis of one of the workshops to be run at the **TRANSPUTING 1991** conference in

California in April 1991, and it is hoped that the experimental details (and maybe some of the solutions) will be published, too. More details are available from the chairman.

Roger Peel +44 483 509284
Department of Electrical Engineering roger@uk.ac.surrey.ee
University of Surrey
Guildford
Surrey GU2 5XH
United Kingdom

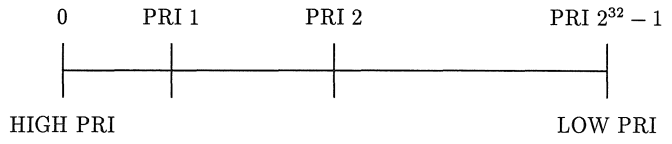
REAL-TIME SIG

André Bakkers, University of Twente

Some forty-five of those interested in real-time systems attended this first meeting, at the OUG meeting in Exeter, led by the self-elected chairman Andy Bakkers of Twente University. Andy's background is in the (robot) control area and as such he defined the term real-time systems as: 'in real-time systems the correctness of a computation does not solely depend on the correct logic but also on the time at which the results are made available.' He introduced the use of priorities in an extended 'Xoccam' as:

```
PAR
  PRI 1:
  ....process.1
  PRI 2:
  ....process.2
  ....process.3  ;process.3 takes on same priority as process.2!
```

On a scale it looks like:



With these principles the two 'standard' scheduling algorithms may be implemented as follows:

MONOTONIC RATE ALGORITHM This algorithm is the most simple to implement because it uses fixed priorities equal the the repetition time period of the process. Its disadvantage is that scheduling can only be guaranteed if utilization is kept under 70% although some scheduling may work beyond this point. Say process i has the priority P_i equal to period i. The Xoccam program could look like this:

```
PAR
  PRI P1:
    proc1 ()
  PRI P2:
    proc2 ()
```

DYNAMIC DEADLINE ALGORITHM This algorithm lets you schedule up to 100% utilization minus overhead. This algorithm poses an implementation problem because the time all goes in the sorting of the priorities if done in software. Here the process *i* has a priority D_i that is equal to the deadline time of *i*, the changing deadline will be computed elsewhere in the program. The equivalent Xoccam program looks like:

```
PAR
  PRI D1:
    proc1 ()
  PRI D2:
    proc2 ()
```

Next Klaas Wijbrans also from Twente University explained what could be done in real-time scheduling using occam. He proposed that the real-time process polls a high-priority scheduler process that grants or denies permission to continue. The real-time process has to be modified to include at regular intervals input from the high-priority scheduler process. Each real-time process is connected to its own scheduling channel.

Peter Welch suggested a discussion on the hooks required for this method to determine what has to be done at the de-scheduling operation. How frequently does the polling have to be done? What is the syntax to use for setting the priorities of processes? Andy stated that polling was not necessary because the pre-emption points are known exactly, i.e. after high priority sampling actions or at completion of the real-time process in question.

Klaas continued to show some results of this method indicating that with a set of three processes and a sample frequency of 500 Hz a maximum processor utilization of 50% could be obtained. A 1000 Hz sampling frequency always resulted in missed deadlines. The maximum possible processor utilization reduced to 30% with four processes at 500 Hz. With these figures this type of scheduler can not be classified as a high performance scheduler, but it is probably the only way a scheduler may be implemented in pure occam.

ROGER SHEPHERD: What about the sharing of priorities between processes that communicate? ANDY: They have to be the same if they are connected otherwise an overwriting buffer should be used. DENIS NICOLE: so use one TP per priority level... PETER WELCH AND OTHERS: ...but you cannot because they have to share data.

The question was raised on the interaction between link copies and high priority processes. ROGER: That's hard although link communications do at present have the priority of the causing process, there still is the problem of the time it takes to complete an atomic action.

Will we have a do-it-yourself scheduler on the H1? ROGER SHEPHERD: Not on a product that we're near designing. The trouble is that the market wants what it knows, not what is elegant; so they're looking at industry-standard real-time kernels. In the H1 high-priority processes will be able to schedule low-priority processes cleanly and the overhead will be much lower (about one-tenth).

Peter Welch in closing proposed a one-day seminar/workshop be held by the Real-Time SIG. Consultation with Hussein Zedan revealed that the theme for the September 1990 OUG meeting in York will be real-time systems so let's all participate actively in that meeting.

Ir. A. W. P. Bakkers
Twente University
EL-BSC Dept.
P.O. Box 217
7500 AE Enschede
Netherlands

Tel: +31-53-892794
(secretary) +31-53-892790
Fax: +31-53-354003
Telex: 44200 thtes
elbscbks@utwente.nl
elbscbks@henut5.bitnet

FORMAL METHODS SIG

Michael Goldsmith, Formal Systems (Europe) Ltd

Through the mysterious inner workings of the User Group management, I appear to have emerged as the new Formal Methods SIG chairman, in succession to Bob Stallard. I am sure that members of the SIG will want to join me in thanking him for his efforts in leading the group over the past years.

A baker's dozen of people attended the meeting, which was held during the 12th OUG Technical Meeting at Exeter in April. There was a quite cosmopolitan mix among the attendees (five countries, and five industrial organisations as well as academics).

The headline news (such as it was) was that I have recently left the Programming Research Group at Oxford University, and am now working (together with Tony Cox, another member of the original occam Transformation System project team) for a small company dedicated to bringing the fruits of the PRG's research to the marketplace. The resident members of its Board are drawn from the academic staff of the Department (including Bill Roscoe, who has led the development of semantic descriptions of occam), and other PRG researchers are available as consultants. Formal Systems has a strong commitment to the promotion of formal methods, especially in the area of concurrency, and is keen to encourage the activities of this SIG (though I trust we can find sufficient enthusiasm from other quarters to prevent this forum degenerating into a continuous commercial for the company).

The other activity which was admitted to was again from Oxford: Geraint Jones has been working on formal design and verification of digital signal processing chips at the PRG. While this is hardware rather than software, the same approaches are applicable in both fields, and his work draws on algebraic techniques originally developed for reasoning about functional programming. He announced a workshop on Designing Correct Circuits to be held in Oxford on 26–28 September 1990.

There followed a discussion of tools for the support of formal development. Geraint expressed some scepticism about their practicality, except perhaps within specific paradigms, such as systolic array design. I agreed that general purpose tools were probably still some way off being a commercial proposition: even a robust and efficient version of the occam Transformation System would be prohibitively costly in the demands it would make on its operator to gain wide acceptance. Formal Systems instead plans to derive special purpose transformations from our experience with the more general engine, to meet clearly perceived needs; some suggestions were forthcoming as to what these might be, with recursion-elimination aids coming high on the wish list. Jaap Hofstede reported a degree of success at the University of Twente in transforming Lotos to occam, via an intermediate restricted subset of Lotos, with only limited manual intervention.

The rest of the session dwelt mainly on possible extensions to occam, and their impact on formal reasoning. Dynamic memory allocation is one thing that programmers are crying out for (or voting with their keyboards and writing C); but Denis Nicole of Southampton pointed out that there are PADs which come with a firmware bug-list some inch thick, mostly due to failing to deallocate memory correctly. The symptoms are that the embedded system runs perfectly for months, until the slow erosion of heap space causes it to crash – not a desirable feature in, say, a fly-by-wire controller.

occam language SIG?

With a number of derived versions of the language possibly in the offing – a short-term *occam 2-and-a-bit* and a longer-term *occam 3* rumoured from Inmos; a standardised occam for verified systems to be proposed by the Esprit PUMA collaboration; and variants like Southampton's H1-routing emulation compiler – there was some support for the idea of a new SIG on occam language issues. It may be that the first of these languages will already have been fixed before the next meeting in York, but the others still warrant discussion. I will be happy to act as a clearing house for expressions of interest in such a SIG; would anyone who would like to join one (especially volunteers for the chair!) please contact me.

I opened the meeting with a statement of good intentions: to try and make more happen in the Formal Methods SIG; I close the report with the reminder that, in the way of the world, things are much more likely to occur if I am given encouraging prods from time to time, or even constructive suggestions. Anyone who can come up with either is most welcome to contact me.

Michael Goldsmith
Formal Systems (Europe) Ltd
Unit 7, The S.T.E.P. Centre
Osney Mead
Oxford OX2 0ES
United Kingdom
Tel: +44 865 728460
Fax: +44 865 793165
michael@uk.ac.oxford.prg

HARDWARE SIG

Denis Nicole, University of Southampton

A meeting of the hardware SIG was held in the morning of 4th April 1990 during the Exeter OUG meeting. As usual, the meeting consisted of several short presentations about items of current interest.

New designs from Division

Charles Grimsdale described some new TRAM modules designed by Division Ltd in Shepton Mallet. They are being marketed by Transtech.

- ▷ The Static Column RAM (SCRAM) TRAM uses fast page mode dynamic memory to achieve two cycle access on page hits by a T801-30 transputer. There is only one active page, of 2 kbytes. The TRAM also has 32 k to 128 k of static memory. In typical use, vector data is held in the DRAM and code in the SRAM. The latter is essential to prevent page misses (and loss of the active page) due to code access. From 2 M to 8 M of DRAM may be fitted on the size two TRAM. Only two PALs are used to manage the column mode access; they also provide a fast vector initialization facility, zeroing memory at up to 30 Mbyte/s. A performance of up to 4.3 Mflop/s is claimed for suitable vector codes. (See page 80)
- ▷ The i860 TRAM, intended for fast image generation, boasts a 40 MHz i860 processor, a T805-25 transputer and 20 Mbyte of dynamic memory on a ten layer size six TRAM board. The memory is organized as 16 Mbyte shared between both processors and 4 Mbyte private to the transputer. Standard 70nS DRAM is used; on the transputer side the private memory is accessed in four cycles and the shared memory in five cycles. This shared memory is organized in two banks, interleaved bank accesses run in four cycles. On the i860 side, a single wait state is required for page hits, two wait states for memory to memory copies. Posted writes and pipelined addressing are supported. Inter-processor synchronization is achieved by using interrupts on the i860 and events on the transputer. In typical use, the i860 is run as a coprocessor for the T805, which runs conventional transputer code. The i860 is claimed to be a factor of 10 to 12 faster than the transputer on most codes and to have a performance of 3.4 Mflop per second per square inch. All control functions are achieved in twelve PALs.
- ▷ Charles also admitted to a rumour of a high-speed 200 Mbyte/s point-to-point interprocessor communication system under development with Transtech.

Inmos comments

Roger Shepherd discussed the need for a new TRAM format for the forthcoming H1 transputer and its new links. He pointed out that the existing TRAM format has made for considerable competition between suppliers, forcing down end-user prices.

Roger also explained that the core of the T400 transputer will be made available for ASIC designs; Inmos customers will be able to design their own special purpose transputers.

Netbus Developments

Adrian Lawrence of Oxford University described the new H1 compatible Netbus design. This used a three wire bus (serial data, message boundary, 5 MHz clock) to implement standard system services such as link speed selection, reset, error reporting, programmable logic bootstrap etc. Futurebus levels are used for the bus signals. A Netbus slave can be implemented in a single Texas PSG507 chip. Adrian is currently specifying the bus protocols in the Z specification language; he will then refine this specification to a logic design in Couple.

Hardware Standards

James Kidd of Altis offered to conduct a survey of transputer hardware standards and publish the appropriate interface specifications. His offer was gratefully accepted by the SIG. Copies of the questionaire may be obtained from

James Kidd
ALTIS
9 Brackley Street
Warrington WA4 6DY
United Kingdom

Denis Nicole +44 703 787167
University of Southampton dan@uk.ac.soton.ecs
Department of Electronics and Computer Science
The University
Highfield
Southampton SO9 5NH
United Kingdom

ENVIRONMENTS SIG

Peter Clare, THORN EMI Central Research Laboratories

The recently formed Environments SIG held a meeting at the 12th OUG meeting in Exeter. The Environments SIG was formed from the Operating Systems SIG and the UNIX SIG in order to cover the key area of all programming environments within a broader perspective.

I had stepped in at the last minute to chair this Environments SIG meeting. So, I opened the meeting by first introducing myself as the Technical Project Manager of the ESPRIT funded Supernode II project. I then tried to explain the background to the formation of the SIG. This was difficult because I had only been to one previous meeting of the UNIX SIG! I looked to the rest of the meeting for help, but there seemed to be a similar lack of continuity between the old Operating Systems and UNIX SIGs and those present.

I ventured to suggest that a title of 'Operating Systems and Environments' would be more appropriate for this SIG. This more descriptive title might have attracted more than the twenty-five (or so) people who attended the meeting. There was neither a positive nor a negative response to this suggestion. Coincidentally (perhaps) the title of the Supernode II project is 'Operating Systems and Programming Environments for Parallel Computers.'

We started the technical discussions by talking about operating system standards and how they relate to the existing operating systems for transputers. In the operating system world we have a two horse race. UNIX System V release 4 would seem to be the current front runner, with OSF/1 a late starter. Mach looks increasingly promising in the area of distributed operating systems (and has been given a recent boost by OSF). In Europe, Chorus is probably the favourite amongst various distributed operating systems. Where does all this talk of standards leave the user of transputer operating systems such as MeikOS, Genesys, HELIOS or Idris?

A user of MeikOS (who shall remain nameless) stated that if there was to be a standard transputer operating system then it should not be MeikOS! H. Thomas of Transtech sang the praises of Genesys. Unfortunately, there were no users of Genesys in the audience to back up the claims. A number of users of HELIOS spoke of bugs and problems, but generally, ways around many of the problems had been found. Nigel Bamford of THORN EMI said that early versions of Idris had been slow, but this had improved recently. Amongst those present, there was no clear favourite operating system, although HELIOS was probably the one most widely used. Some of these operating systems are attempting to follow UNIX standards by claiming POSIX conformance.

A more general discussion ensued which touched on the following main topics: portability, programmability, diagnostic tools, memory protection and fault tolerance.

Standardisation of object file formats for compiled languages was identified as a particular cause for concern. Inmos have announced a new object file format for all their future compiler products. With the backing of Inmos, will this oust the current Inmos/3L standard? This provoked a lively discussion between Michael Poole of Inmos and John Fisher of 3L. The current 3L position is that they will not change their compilers to generate code in the new format. The meeting felt that it was a pity that, in this area, standards were diverging rather than converging.

Overall, we felt that the meeting would have benefited from the presence of representatives from Meiko, Parsys and Perihelion to answer questions on their respective operating systems. Let us hope that we can attract such representation at the next meeting in York.

P. J. C. Clare clare@uk.co.thorn-emi-crl
Thorm EMI
Central Research Laboratories
Dawley Road
Hayes
Middlesex UB3 1HH
United Kingdom

GRAPHICAL PROGRAM DEVELOPMENT TOOLS SIG

Mike Roberts, City University, London

The first meeting of the Graphical Program Development Tools Specialist Interest Group meeting was held at the last OUG technical meeting. About 30 people attended the evening meeting. Notably, a strong contingent from industry attended, perhaps indicating that market forces may be moving in the direction of graphical tools.

A lively discussion with an almost ‘evangelical’ feel ensued, with members of the community standing up to tell the meeting about graphical tools in development, proposals for new tools, and summaries of existing tools.

Mechanical Intelligence told us briefly about the graphical configuration facilities of Express. Express seems to be one of the few tools with graphic configuration facilities in widespread use.

The benefits of Susan Stepney's GRAIL graphical performance monitoring tool were also extensively discussed, with many people feeling that GRAIL required further development as a product for the entire transputer community (not just those with T-racks). Undoubtably such an effort would be well repaid. Interest in just such a tool was expressed by Jonathan Gulley from Thorn–EMI who instigated an interesting discussion by canvassing attendees on their views as to what a graphical program visualisation/performance monitoring tool should consist of. Many views were put forward, but a consensus was reached on the issue of portability – any such tools should be portable over a wide range of transputer based architectures and host systems.

N. Winterbottom (IBM) told the meeting of his experience with visualisation tools based on simple calls inserted into code calling display routines. Such tools are apparently not only easy to implement and use but also give a good return on the effort taken to implement them.

Other tools discussed included City's visual programming tool GILT, the Gecko configuration tool from PCL and the work at Manchester on the automatic generation of occam PROC headers from a process diagram drawing package.

Proposals for discussion at future meetings included a one day workshop on graphical tools, a more formal basis to the group, and possible demonstrations of tools at future meetings.

Anyone attending the meeting has been 'conscripted' into a list of SIG members – anyone else wishing to voluntarily 'join up' should contact me (preferably by email).

In conclusion then, a successful first meeting of the SIG with a bright future in view.

Mike Roberts +44 71 253 4399 x3889/3877
The Centre for Information Engineering m.roberts@uk.ac.city
City University
Northampton Square
London EC1V 0HB
United Kingdom

TECHNICAL CONTRIBUTIONS

RESTARTING THE TDS LINK MULTIPLEXER

Steve Gillanders, Keydata S.A., Buenos Aires

I would like to report a trick regarding the use of the TDS *link0* multiplexer/ demultiplexer.

Suppose you, as we did, modify the source for the the TDS link multiplexer adding a couple of extra channels for graphics and specialized serial communications and filing services. You also modify the TDS service on the PC side to accept the new channels and process the messages for graphics, etc.

Now it will be possible to write EXE programs that use these new capabilities by adding the new mux/demux as a process parallel with the rest of the application.

This allows us to preserve the TDS loader as originally provided by INMOS, supposing we want to minimise the extent of the modifications to be made. Also we need the new multiplexer source available for debugging purposes.

The only problem is that this new *link0* multiplexer process conflicts with the one provided by the TDS by using *link0*. Output is no problem since once the EXE has started up we can be sure that the original TDS mux is not outputting. Input though is a problem: the original mux is sitting at a `PRI ALT` between the stopper channel and the *link0* input.

Fortunately overwriting the *link0* input channel word by executing our new link multiplexer has no other effect than the one intended. All messages coming in through the link are processed by our waiting link demux. The problem is now EXE termination.

When we terminate our EXE, we have beforehand terminated the new link mux/demux process. Again the *link0* output channel is no problem, but we have now ceased activity on the *link0* input. Our system now hangs, since messages from the PC server side are not received by a corresponding process in the transputer because the *link0* input word says `EMPTY`.

Our first attempt at a solution was: OK, preserve the workspace address in the *link0* input word before starting the new mux/demux process, then before terminating the EXE, but after terminating the new mux/demux, copy this saved address back into the *link0* input word. No luck, the system still hangs.

After some experimentation, we arrived at a solution: enable the *link0* input word as if you were in the original demux ALT. Eureka! it works. Figure 1 shows the necessary code.

Steve Gillanders
Keydata S.A.
Crisólogo Larraide 1801
(1429) Buenos Aires
Argentina

Tel: +54 1 70-4467/3281
Telex: 23096 KEYSA AR
Fax: +54-1-11-2426

IF TRANSPUTERS AND OCCAM RELIABILITY := HIGH

Some arguments about fault-avoidance and fault-tolerance

L. Borisov and G. Topping
Computing Department, Staffordshire Polytechnic

The chairman of the occam user group did well in his article in the January 1990 edition of this *Newsletter* [1] to underline the growing importance of reliability and safety in computing, and to emphasise the valuable properties of the transputer and occam in safety critical applications. His remarks focused upon the fact that both occam and the transputer are based upon formal mathematical theories of parallel computation and emphasised the firm foundation this provides for the production of safety critical software. Occam provides not just another programming language but also a vehicle for expressing parallel algorithmic specifications which, with any degree of formality one chooses, may be transformed into executable form with great advantages in terms of software reliability and provability.

All these stated features of the transputers and occam actually provide a means to

```
INT link0in, old.boot.contents :
PLACE link0in AT 4 :            -- link0 input word
SEQ
  old.boot.contents := link0in  -- save workspace in channel
  PAR
    .. new link mux/demux process
    ... application
  GUY
    LDL old.boot.contents -- saved workspace for mux/demux ALT
    LDC #FFFFFFFE         -- careful with priority bit
    AND                   -- eliminate low priority setting as it
                          -- will be added later
    GAJW                  -- switch to ALT's workspace
    STL 0                 -- store EXE workspace temporarily
    MINT
    ADC 16                -- form link0 channel address
    LDC 1
    ENBC                  -- enable link0 input with ALT workspace
    LDL 0                 -- load EXE workspace
    LDC -1
    STL 0                 -- set ALT workspace word 0 back to -1
    GAJW                  -- switch back to EXE workspace
```

Figure 1: Code to restart the TDS *link0* multiplexer

prevent the occurrence of faults, so we consider these capabilities as fault-avoidance support. But there is another reliability aspect which we would like to highlight – the fault-tolerance aspect: the use of protective redundancy to permit continued correct operation of a system after the occurrence of specified faults.

Many well known fault-tolerant computers, such as the Space Shuttle Computers [2], FTMP (Fault-Tolerant Multiprocessor) [3], SIFT (Software Implemented Fault-Tolerance) [4] are based on parallel processing using multiple (redundant) processing elements. This parallelism is deployed in the design of their hardware voting schemes and also in the use of 'pair and spare' techniques [5]. It is clear that a transputer array can be used to satisfy such redundancy requirements and occam can be used to express explicitly the necessary parallelism.

In examining these classical examples of fault-tolerant systems we notice the use of replicated (redundant) bus systems in order to provide for more reliable communications. There is no need to design and implement such a replicated communication mechanism for transputers because it already exists – we have four links per transputer. In a point-to-point communication system a link failure can be tolerated using timeout techniques and a graceful performance degradation can be achieved. The transputer link communication offers a ready tool for exchange of information with fault-tolerant schemes based on test-assignment and voting, and also facilitates mutual watchdog observation (for example in detection of dead nodes).

The need for synchronization when exchanging information is met by the link hardware. This synchronized link communication automatically provides memory write protection from a faulty processor. This kind of protection against corruption of data has been achieved by special hardware design in the SIFT computer [4], where a processing module can read data from any processing module's memory, but it can write only into its own memory.

Transputer-based solutions for achieving high reliability have already been proposed [6, 7, 8] and show the considerable potential of the transputer in the fault-tolerant field. This potential supplements very well the capabilities of the transputer for real-time and embedded applications.

In the course of his argument for *Safety First* [1], Peter Welch asked rhetorically 'Can an i860 outperform four T800s?' Here is our version of the answer. Let us assume that one i860 is twice as fast as four transputers, but the transputer array with distributed memory runs its program in a manner which permits graceful degradation in the presence of faults. A single hardware fault in a memory or processor chip can easily slay the i860 giant whilst the 'outperformed' transputers will continue to provide service.

The use of more i860s will require at least three chips to provide fault recovery and we will immediately face increased costs, additional design problems to provide redundant communication systems and the synchronized exchange of information, and the use of inappropriate languages for parallel software development. Whilst admiring the capabilities of the i860 and its like, we consider the transputer as having its own particular advantages if performance is considered to include not merely speed of execution but also resilience to errors and faults.

As their dependence upon computers increases, will the public at large be content with an industry which equates performance with speed? We think not. No amount of gigaflops can compensate for even a microfault in sensitive applications whose failure could threaten life and property. That is why the emphasis in system development needs to shift towards failure tolerant design. The transputer is, in many ways, ideally suited to the special techniques required and there is no reason why those techniques cannot be applied much more widely than at present. Indeed the transputer is a suitable candidate for realising the low cost non-stop computing platform.

We echo Peter Welch's conclusion that the pressure for safety is likely to outrun the pressure for performance narrowly interpreted. Like him too, we ask why Inmos publicity does not promote its product with this in mind and, in doing so, additionally highlight the claims for the transputer in fault-tolerant design.

References

[1] P. Welch, *Safety First, Performance – a Secondary Consideration*, Occam user group newsletter Nº 12, January 1990. pp. 22–27

[2] J. R. Sklaroff, *Redundancy Management Technique for Space Shuttle Computers*, IBM Journal of Research and Development, January 1976. pp. 20–28

[3] A. L. Hopkins and T. B. Smith, *FTMP – A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft*, Proc IEEE, October 1978. pp. 1221–1239

[4] J. H. Wensley, L. Lamport and J. Goldberg, *SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control*, Proc IEEE, October 1978. pp. 1240–1255

[5] R. Freiburghouse, *Making Processing Fail-Safe*, Mini-Micro Systems, May 1982. pp. 255–264

[6] H. A. Thompson, P. J. Fleming and C. G. Legge, *Implementation of a Fault Tolerant Transputer-Based Gas Turbine Engine Controller*, Workshop on Real-Time Systems – Theory and Applications, 28–29 September 1989, University of York.

[7] J. Standeven and M. J. Colley, *Performance Aspects of Transputers in Fault-tolerant systems*, Workshop on Real-Time Systems – Theory and Applications, 28–29 September 1989, University of York.

[8] L. Borisov, G. Topping and F. Wright, *Using Transputers in Fault-Tolerant Real-Time Control*, Conference on Reliability in Computer Science, 9–13 April 1990, Jasna pod Chopkom, Czechoslovakia.

L. Borisov
Computing Department
Staffordshire Polytechnic
Blackheath Lane
Stafford ST18 0AD
England

Tel: +44 785 53511 x 6053
cdrlb@uk.ac.stafpol.cr83

SEMAPHORES AT THE TRANSPUTER INSTRUCTION LEVEL

Dieter Homeister, Universität Stuttgart

The Transputer has no special undividable test-and-set instruction like those of the 68000 and other processors, which can be used for the implementation of semaphores at the instruction level. The only instruction that reads and modifies a memory word is `RESETCH`, which was designed to cancel a deadlocked channel communication. The function is described in references [9, 10]. The memory word which represents the channel can be in one of two states. If the channel is busy, the memory location contains a pointer to the waiting process to be continued after the end of the communication. Otherwise, if the channel is idle, the value of the channel word is `MININT` (hex 8000000 on 32-bit transputers). This constant is called '`NotProcess.p`' and stands for 'no process is waiting and the channel is free'. Before execution, the A-register points to the channel word. The `RESETCH` instruction reads the old contents of the channel word and puts it into the A-register, a value `MININT` is written into the channel word. The value retrieved is compared with `MININT`: if they are not equal the value is a pointer to a waiting process, and this process can be restarted.

This is in fact an undividable read-modify-write instruction that can distinguish two states: `MININT` and other values. This can be used to implement a very fast binary semaphore. The only side effect is that if the affected memory location is a hardware link address the DMA controller is reset, so the semaphore must not be placed on a hardware link location!

Binary semaphores are equivalent to general (counting) semaphores. This conjecture of Dijkstra's is proved in reference [11]. Counting semaphores may be imple-

```
...
semaphore = 0    -- initialise before use
PAR   -- (only 1 of 2 or more processes shown)
  ...
  try_again:
  tmp = RESETCH (address of semaphore) -- try to set the semaphore
                                       --   tmp is a temporary variable
  if (tmp = MININT)                    -- if in use by an other process
      deschedule_this_process          --   wait
      jump try_again                   --   and try again later
  --             -- the semaphore is set now
  begin_of_critical_region
  ...
  end_of_critical_region
  semaphore = 0           -- release semaphore
  ...
```

Figure 2: busy implementation fo a semaphore

mented within a critical region, which is protected by a binary semaphore.

There are some other possibilities to implement semaphores, but they have disadvantages. A channel implementation consumes more time, and it is hard to share a semaphore between more than two processes. Manipulating the internal process queues is not supported by INMOS and may be changed in future. Some implementations use extra process queues, but this is slower and more expensive.

A busy implementation

The example in figure 2 shows the implementation of parallel processes using a semaphore to protect a critical region in a structured assembler-like notation. It has been tested in parallel C, but it may be used in occam too. The test of the semaphore takes only a few CPU cycles, if no other process blocks the semaphore. If the semaphore is not free, the process waits until the other process releases the semaphore. The waiting in a loop looks busy but for most of the time the waiting process is descheduled and consumes no CPU time. The semaphore is fair because there is no static priority. The semaphore must be initialised with a value not equal to MININT. All used variables are word-sized.

The following assembler instructions [12, 13] are used to deschedule a process immediately and to put it at the end of the process queue, so that the next active process is able to run. This is independent of the current priority.

LDLP 0	–	pointer to the workspace of this process
LDPRI	–	get current priority
ADD	–	set the LSB according to the priority
RUNP	–	add this process at the end of the process queue this process lives twice for a short moment, but only one of them is active
STOPP	–	terminate the running process

```
...
semaphore = 0            -- initialise before use
wakeup_address1  = MININT -- initialise all wakeup addresses
...
wakeup_address_n = MININT
PAR   -- (only 1 of n processes shown)
  ...
  try_again:
  tmp = RESETCH (address of semaphore)   -- try to set the semaphore
  if (tmp = MININT)                      -- if used by another process:
      LDLP 0  -- pointer to workspace (*)
      LDPRI   -- priority             (*)
      ADD     -- set priority bit     (*)
      store to wakeup_address1        (*)
      STOPP   -- stop process         (*)--   wait (non-busy wait)
      jump try_again                     --   and try again later
  critical_region
  tmp = RESETCH (wakeup_address2) -- is process 2 waiting
  if (tmp <> MININT)              --   for the semaphore?
      RUNP (tmp)                  --   if so, wake it up
    .
    .
    .
  tmp = RESETCH (wakeup_address_n) -- do this for all other processes
  if (tmp <> MININT)
      RUNP (tmp)
  semaphore = 0            -- release semaphore
  ...
```

Figure 3: non-busy iplementation of a semaphore

In high priority processes this can be replaced by '`LDLP 0; RUNP; STOPP`', and in low priority processes '`LDLP 0; ADC 1; RUNP; STOPP`' may be used.

The above semaphore construction runs with two or more processes in high, low and mixed priority. In case of high priority processes the user must be careful to break the high priority processes from time to time by input, output or timer operations otherwise the low priority processes are unable to run.

A non-busy implementation

A non-busy wait implementation for more than two processes is the 'wakeup' (shown in figure 3) with one wakeup address per process. At the end of the critical section the process has to test the wakeup addresses of all other processes, and restart this process if it was stopped. The test takes longer the more processes are participating, so the first implementation seems to be better even if the waiting is not totally non-busy.

The statements marked with a star n figure 3 may be replaced by a dummy `IN` statement, which is equivalent.

```
-- producing process
...
produce_some_data
increment data_produced
process_id = RESETCH (wakeup_address) -- is the consumer sleeping?
if (process_id = MININT)
      RUNP (process_id)              --   if so, wake it up
...

-- consuming process
...
loop:
if (data_produced <= data_consumed)   -- unprocessed data available?
    IN (wakeup_address)               --   if not, sleep
    -- this dummy IN has no partner OUT, so it hangs until
    -- the producer process executes RUNP
-- if data is present, there is no delay
consume_some_data
increment data_consumed
goto loop
...
```

Figure 4: fast synchronisation using communication on only half a channel

Fast synchronisation

This technique of communitcating on only one side of a channel is also useful for implementing a fast synchronisation if a consuming process has to wait for data, see figure 4. If the data is already present the process continues without descheduling. The two counters *data_produced* and *data_consumed* are shared variables. They are ised carefully: each variable may only be written by one process; both processes can read them. This is allowed only if the increment statement produces code without descheduling points, and if both processes run at the same priority.

References

[9] INMOS Limited, *The transputer instruction set – a compiler writers' guide*, Bristol, November 1987.

[10] Roger Shepherd, *Extraordinary use of transputer links*, INMOS Technical note 1, Bristol, 1987.

[11] Jürn Jürgens, *Synchronisation paralleler Prozesse anhand von Zuständen*, Ph.D. Thesis, Munich, 1973.

[12] Andreas Kaiser, *Assembler und Linker für Transputer*, Studienarbeit, University of Stuttgart, 1989.

[13] Andreas Kaiser, *Parallel C Compiler für Transputer*, Diplomarbeit, University of Stuttgart, 1990.

Dieter Homeister
Universität Stuttgart
Institut für parallele und
verteilte Höchstleistungsrechner
Azenbergstaße 12
7000 Stuttgart 1
West Germany

Tel: +49 711-121-1342
Fax: +49 711-121-1424
homeister@informatik.uni-stuttgart.dbp.de

MORE ON THE IMPLEMENTATION OF SEMAPHORES IN OCCAM

S. W. Lau and F. C. M. Lau
Department of Computer Science, University of Hong Kong

In the last issue of this *Newsletter*, Geoff Barrett suggested some ways of implementing semaphores in occam [14]. His methods cater both for processes of only one priority and for processes of two priorities. The implementation, however, requires the use of some extended language features which are not present in the standard

```
VAL NULL IS #80000000:
[3]INT s: -- the semaphore
-- s[0] = s.v, s[1] = head of queue, s[2] = tail of queue

PROC init([]INT s, INT n)
  s[0], s[1] := n, NULL
:

PROC wait([]INT s)
  IF
    s[0] > 0
      s[0] := s[0] - 1
    TRUE
      SEQ
        ... insert Wptr of process into the queue
        ... stop execution
:

PROC signal([]INT s)
  INT wptr:
  IF
    s[1] = NULL -- queue is empty
      s[0] := s[0] + 1
    TRUE
      SEQ
        ... wptr := Wptr of the first waiting process in the queue
        ... run the process with wptr
:
```

Figure 5: outline of the implementation of semaphores

occam. In fact, we found no way of putting a channel into a list with the current implementation of occam. Nonetheless, with the help of transputer instructions, we were able to come up with an actual non-busy, efficient implementation of semaphores on the transputer. We did it only for the case of single-priority processes, which we think is sufficient to illustrate our idea; using the solution suggested by Barrett, the code can be easily extended to work for processes of two priorities.

Our Implementation

Our idea is the obvious one: instead of keeping a FIFO queue of channels upon which processes queued for a semaphore would be waiting we keep a FIFO queue of processes by putting their workspace pointers [15] into the queue. Whenever a process issuing a *wait* on a semaphore is blocked, a pointer to its workspace is inserted by transputer instructions into the FIFO queue associated with the semaphore. A *signal* issued by an active process would wake up the first waiting process – if there is one – in the queue. Figure 5 shows the outline of our implementation for processes of one priority. Figure 7 contains the full listing of the implementation. To maximize performance some of code which was originally in occam has been replaced by transputer instructions.

The experiment

We used the example suggested in Barrett's paper to compare the performance of TDS's implementation of ALT and our semaphore implementation. In this example N processes continuously send data to a common receiver process. The program terminates when every process has finished sending out (in this case) the 1000 integers it is supposed to send. For measuring the performance of ALT, the following code was used:

```
PAR
  SEQ i = 0 FOR N * 1000
    ALT j = 0 FOR N
      c[j] ? x
        SKIP
  PAR i = 0 FOR N
    SEQ j = 0 FOR 1000
      c[i] ! 0
```

For measuring the performance of the semaphore, the following code was used:

```
PAR
  SEQ i = 0 FOR N * 1000
    c ? x
  PAR i = 0 FOR N
    SEQ j = 0 FOR 1000
      SEQ
        wait(s)
        c ! 0
        signal(s)
```

N	semaphore	TDS's ALT
1	7,618	8,744
5	57,609	143,500
10	115,227	545,799
50	584,155	13,489,449
100	1,089,652	54,873,263

Figure 6: comparison of execution times

In each case the total time required for executing the code was measured, from which was then subtracted the time (measured in a different experiment) required for looping, communications, and context switching. All measurements were done on an inmos B404 TRAM, using TDS (D770D). In each measurement the execution code was put into the TRAM's DRAM which has an access time of two clock cycles. All measurements have a two microsecond accuracy. The results (in microseconds) are tabulated in figure 6 and show that our implementation of semaphores outperforms TDS's implementation of `ALT` in all cases. This is expected since `ALT` is known to be rather slow especially when the number of alternatives is large [16]. From the results, the time for executing a *wait* and a *signal* is approximately 10 microseconds (not including the communication in between). For the case of $N = 1$, since the sender process is never blocked, no insertion into and removal from queue is needed, and the time is less.

Conclusion

We have described an easy but efficient implementation of semaphores for processes of a single priority on the transputer. The timing results suggest that our implementation is efficient enough to replace `ALT` in some forms of alternation – those that are typical of many-to-one communications between a server and its clients (in fact, this kind of communications is believed to be so popular that the future H1 transputer will commit the *wait* and *signal* instructions onto the transputer instruction level as well as introduce a many-to-one channel instruction [17]).

References

[14] G. Barrett, *Two implementations of semaphores in occam*, OUG Newsletter, Nº 12, January 1990.

[15] Inmos Ltd, *Transputer instruction set – a compiler writer's guide*, Prentice-Hall, 1988.

[16] K. M. Shea and F. C. M. Lau, *On the performance of* `ALT` *in occam*, NATUG 3, April 1990.

[17] D. Pountain, *Virtual channels: the next generation of transputers*, Byte, Vol. 15, Nº 4 (E&W section), April 1990.

```
VAL NULL IS #80000000:

-- s.v -> s[0]
-- s.head -> s[1]
-- s.tail -> s[2]

PROC init([]INT s, VAL INT n)
  -- s.v, s.head := n, NULL
  s[0], s[1] := n, NULL
:

PROC signal([]INT s)
  INT wptr:
  GUY
    LDLP s
    LDNL 1
    MINT
    DIFF
    CJ .equal
    -- remove first element
    -- wptr := s.head
    LDLP s
    LDNL 1
    STL wptr
    -- s.head := next[s.head]
    LDL wptr
    LDNL -2
    LDLP s
    STNL 1
    -- run process wptr
    -- in current priority
    LDL wptr
    LDPRI
    OR
    RUNP
    -- jump exit
    J .exit
    :equal
    -- s.v := s.v + 1
    LDLP s
    LDNL 0
    ADC 1
    LDLP s
    STNL 0
    :exit
:
```

```
PROC wait([]INT s)
  IF
    -- s.v > 0
    s[0] > 0
      s[0] := s[0] - 1
    TRUE
      SEQ
        -- next := NULL
        GUY
          MINT
          STL -2
        -- enqueue
        GUY
          LDLP s
          LDNL 1
          MINT
          DIFF
          CJ .equal
          -- next[s.tail]:=wptr
          LDLP 0
          LDLP s
          LDNL 2
          STNL -2
          -- s.tail := wptr
          LDLP 0
          LDLP s
          STNL 2
          -- jump exit
          J .exit
          :equal
          -- s.head := wptr
          LDLP 0
          LDLP s
          STNL 1
          -- s.tail := wptr
          LDLP 0
          LDLP s
          STNL 2
          :exit
        -- wait
        GUY
          STOPP
:
```

Figure 7: source of the implementation of semaphores

Francis C. M. Lau
Department of Computer Science
University of Hong Kong
Hong Kong

fcmlau%csd@hkujnt.bitnet
fcmlau@csd.hku.hk

ALL ABOUT TROLLIUS

Greg Burns, Vibha Radiya, Raja Daoud and Raghu Machiraju
The Ohio State University

An opportunity for a standard?

Trollius was conceived because the best multicomputer software in existence at the time of its birth, Autumn 1985, was not very appealing even to the adventurous computational scientist. In particular the transputer, building block of a new generation of multicomputers, had essentially no operating system. Trollius started with both a general and a specific purpose.

An opportunity appeared to build a strong operating system foundation suitable for future multicomputers. Contemporary system software from the hypercube generation appeared to be hastily constructed solutions from companies under pressure to reach the market fast and focus attention on the promise of the architecture. Since the first commercial introduction of multicomputers, hardware concerns have been progressively addressed while software has lagged behind. Standards have not appeared. Trollius did not anticipate a multicomputer as a stand-alone workstation, but virtually every other possible usage is supported. New operating systems continue to be built, often for proprietary reasons, and the lowest common denominator remains stalled at the message passing multicomputer architecture. Until higher platforms are accepted, progress will be slow.

The Trollius Project is structured for maximum impact on the young field of concurrent processing. Within the university environment, the development team at The Ohio State University has the latitude to put engineering first and ignore the bottom line. Away from academic departments, the developers are full-time professional engineers, concerned with building an industrial strength product first and tackling research issues second. Trollius is the only operating system portable to any multicomputer incarnation and available in source form. In particular, getting the source into the hands of bright students and faculty at other universities has been its formula for survival. People can use other systems and become expert users. People can perform surgery on Trollius and become guru developers. This article will point out that the structure of Trollius endeavors to make users and new doctors comfortable and uninhibited.

Mission for a perceived supercomputer

Trollius was born, raised and continues to live in the house of US supercomputer centers. From this perspective, the ultimate purpose of scalable parallel architectures is to extend the limits of supercomputing and permit attacks on the grand challenges of computational science. Thus the first concern of Trollius has been to support data parallel applications. Many other system and language solutions emphasize their elegance and efficiency at solving function parallel applications. Trollius has concentrated higher level development on data parallelism, but all the building blocks are in place for any strategy of multicomputer problem solving.

Is executing the designated application the only rôle an operating system must play? If the answer is yes, then a network loader and a communication library might be the final solution, because multicomputer applications are typically not very complex – today. Trollius anticipates that the answer is no. Monitoring, debugging, machine maintenance and machine portability must also be considered. If third-party vendors and university research groups are essential to software progress on new architectures then a gutsy operating system must not set limits and must not attempt to predict the future but instead must concentrate on on the accepted basics. The alternative is more new operating systems starting from scratch instead of the shoulders of others.

The basic activity of a multicomputer is passing messages between processes running on nodes. Other activity is an abstraction. Abstractions are great creative things, but the basics need to be accessible to the programmer in a direct, no-nonsense, complete manner, like the standard TTL NAND gate is to the digital engineer, in case one of the higher level packages does not fit the need. No one abstraction is the answer to every problem solving strategy on the multicomputer architecture. A thorough package of the basics supporting any possible abstraction is the answer.

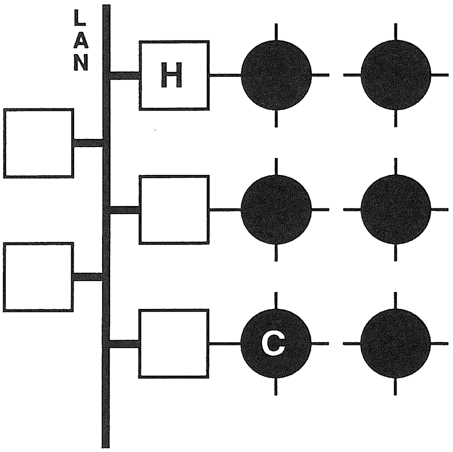


Figure 8: The General Multicomputer – Trollius is a uniform foundation for message passing on every multicomputer node. Host nodes (H) run Trollius with the local operating system while compute nodes (C) run it natively.

The golden rules

The previously discussed motivation and purpose spawned the principles of Trollius's design, listed succinctly below:

- ▹ Focus on supercomputer applications, scalable problems and Unix.
- ▹ Expand the multicomputer scope to include the local area network.
- ▹ Utilize an onion-like design, in which each layer can be peeled off for less functionality but better performance.
- ▹ Always allow the user to access the basics: nodes, processes, links and messages and finally, the hardware.
- ▹ Maintain node uniformity; host and compute nodes run the same Trollius.
- ▹ Engineer modular software that facilitates other development. Structure with multiple processes above a spartan message-passing local node kernel.
- ▹ Don't bet on any one compiler, processor or topology.

The structure and functionality of Trollius

To dispel the suspicion that Trollius is a primitive operating system, the functional description that follows begins at the current top of the evolutionary tree and works downward.

Crystalline (CrOS) is a communication library for domain decomposition developed at the California Institute of Technology. It bypasses the modest, yet annoying problems of grid programming: simple deadlock and global singularities. CUBIX is a file access method, also developed at Cal Tech, that collects and distributes data among multiple nodes in an orderly manner. CUBIX often precludes the need to write a control program for the host or root node. Together, CrOS and CUBIX are the recommended tools for grid domain decomposition; Trollius offers both.

Topology layer

Ready to boot the multicomputer, the user confronts the highest layer of Trollius 2: the topology layer. The topology layer tools interpret a simple grammar called Node and Interconnect Language (NaIL), in which an exact description of the multicomputer to be booted has been recorded. Given this topology description, called the 'boot schema', Trollius endeavours to completely solve the general problem of link configuration, booting and routing with very few limits. In particular topology is not limited. The boot schema describes the number, type (compute node, host, disk connected, CRT connected, wasted interface) and identification (any 32-bit quantity) of multicomputer nodes. The one restriction, removed in Version 3, is that only one host node may exist. Also described are the links between nodes, explicit routes to be taken between two points, priority route assignment between two points, and the correspondence between Trollius node identifiers and vendor-specific naming conventions (so that vendor-specific link configuration tools may be invoked). Most common configurations for a particular installation will be created by the system administrator and saved.

Given multiple links between the host node and the compute node pool, Trollius 2 permits multiple user access to the pool. Trollius does not allow sharing of single links

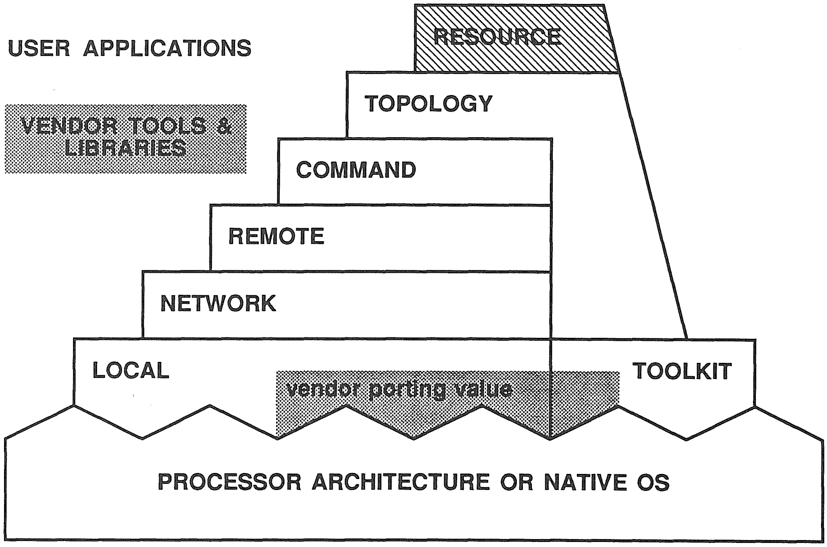


Figure 9: Trollius Layered Structure – Each layer has a standard interface accessible to the user. Vendors can add proprietary value in porting to specific hardware and in bundling tools and libraries. The resource layer is part of the third generation of Trollius.

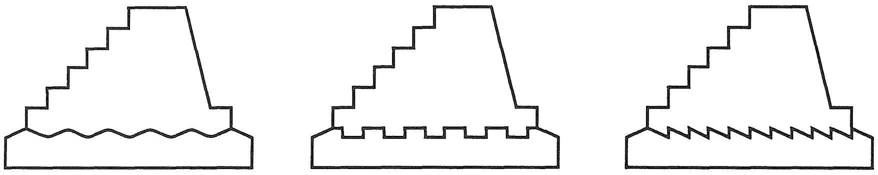


Figure 10: Portable Node Interface – The local and toolkit layers are portable to any compute node architecture. Other layers are unaffected by the underlying environment.

or single compute nodes. Multiple user access in Version 2 is managed by personal communication. This illustrates an important policy of Trollius development – each release selects a cut-off point and provides the complete implementation of the basics to that point. Trollius 1 had no topology layer and the cut-off point for complete solution was the local node operating system. The route table for each node was hard-wired. Booting was accomplished by invoking the individual node booting tools of the command layer – *hboot* (host node), *sboot* (device connected compute node) and *vboot* (remote compute node) – by hand. Today the topology tools drive these command layer tools, and in Version 3 the resource layer tools will drive the topology tools. If a user ever needs to bypass the topology layer, the multicomputer can still be operated from the command layer.

Consider a 2-D problem in computational fluid dynamics solved with a finite difference method on a cartesian grid. Convinced of the merits of concurrent programming, the scientist parallelizes the code using CrOS and CUBIX. He/she has access to a 64 node transputer array connected to a Unix workstation. First, he must compile the program, as demonstrated below:

```
% tf77 -o cfd cfd.f -lcros
%
```

CUBIX is an integral part of the Trollius remote file system and is accessed through the standard library. CrOS is an optional library linked explicitly. This compilation can take place on any Unix machine with an installed and licensed *tf77* cross-compiler supported by Trollius.

The user selects or prepares a boot schema for a multicomputer configuration suitable for the CFD application. Trollius CrOS demands consecutive node identifiers beginning with 0 and continuing through the grid in row-major order. Perhaps the boot schema for an 8×8 grid is called 'bnail64.cros', a slight misnomer because there are 65 Trollius nodes including the host node. The following three topology tools establish a running multicomputer:

```
% map -c bnail64.cros
% solder -c bnail64.cros
% spread -c bnail64.cros
%
```

The *map* step generates a file containing a route table for each node using a greedy algorithm based on accumulating link weights. The weights and more may be fine tuned or over-ridden in the boot schema. Once a route file exists for a given boot schema, the map step can be skipped. Map makes the general problem of routing completely detached from the Trollius Operating System. It is an off-line tool that can be run anywhere. With well-defined input and output files, the creative inventions of individual researchers can replace map.

The *solder* step invokes the vendor-specific software necessary to configure the communication links to match the topology described in the boot schema. Solder is not used with fixed links.

The *spread* step actually boots the multicomputer, invoking the right command layer booting tool for the right type of node and loading route and general network information into each node. Like all Trollius host programs, the user invokes spread from the Unix interface. Control returns to the same interface; no special interface or shell exists. After booting, the user's Unix workstation is also the origin host node in a Trollius multicomputer.

Command layer

A host node program written with the Trollius communication library can cooperate with processes running anywhere in the multicomputer. Trollius supplies control and monitoring programs that make up the command layer. These programs supplement the extensive suite of familiar Unix development tools.

The user is ready to run the program, *cfd*, on the compute nodes using the loadgo command:

```
% loadgo -v C cfd
[1] 348013 cfd running on all compute nodes
%
```

Hands-on information gathering gives Trollius its flavour. For example, is the program running?

```
% state n0,63
NODE    INDEX   PID     KPRI    KSTATE          PROGRAM
n0      [1]     348013  0       A (6, 0)        cfd
n63     [1]     348013  0       A (6, 0)        cfd
%
```

Are the buffers getting clogged?

```
% bfstate -l n32
NODE    DEST    EVENT   TYPE    LENGTH  STATE
n32     n32     6       0       2048    blocked
n32     n32     6       0       2048    waiting

n32     max workers = 4, used workers = 2, full workers = 1
n32     max space = 262144, used space = 4096
%
```

How much memory is free? Perhaps the granularity of the data decomposition can be adjusted.

```
% mstate n7 -f
NODE    PID     ADDRESS         SIZE
n7      free    80000480        B80
n7      free    800535A8        1AD258
%
```

The application either terminates or the user may decide to deliver a termination signal with the *doom* command:

```
% doom C %1
% sweep C
%
```

Killing a message-passing parallel application in mid-flight will often result in a number of unreceived messages in lingering transit. These messages will settle into buffers and can be inspected with the *bfstate* command shown above. Lingering messages can affect the next program run. The debris can be cleaned up with the *sweep* command. Then the program can be run again without rebooting. Afterwards, the host node Trollius is terminated and the compute nodes released by the *tkill* command.

```
% tkill
%
```

The topology and command layers form the operational level of Trollius. The programming interface begins with the remote layer.

Remote layer

Commands process options and present data surrounding one or more invocations to remote layer functions found in the remote library, trillium (Trollius's original name). Thus any user program can accomplish the function of any command program. A remote function exchanges messages with a remote daemon. For example, the command *bfstate* calls the function *rbfstate()*, which talks with the daemon, *bufferd*, on the indicated node.

The daemon called upon most often by user programs is *filed*, which provides a remote interface to the Unix filesystem. Any compute or host node can and run *filed* if it operates a disk. The service includes a large number of standard Unix functions supporting the C stdio package, the Fortran Unit I/O system or direct access of portable Unix programs. (Note: this illustrates another important Trollius policy concerning Unix. Trollius is real Unix where it must be – the host nodes; Unix compliant where it makes sense – the filesystem; and not Unix where it would serve badly – communications.)

Filed only runs on nodes with attached disks. Other daemons only run on compute nodes, where Trollius is the native operating system and has more responsibility for basic services. The following table lists the other remote daemons, their responsibilities, and where they run.

DAEMON	NODE	FUNCTION
mallocd	compute	memory allocation, deallocation and monitoring
kenyad	compute	process creation, signalling, monitoring and spawning
memd	compute	fetch/load specific memory addresses
flatd	compute	load memory and assign tag to address, resolve tags
bootd	compute	boot neighbouring compute nodes
loadd	host	load data and program files on compute nodes
echod	all	echo messages

Network layer

This excursion to the depths of Trollius has reached the layer at which CrOS is written: the network layer. If not the heart, the network layer is certainly the hub of the operating system. A group of processes that run on every node provide a network message-passing service. At the network layer, Trollius functionality is completely uniform among all node types.

The smartest process is the router, which keeps the identifier, type and route to every other node in the system. One will be able to change this information at runtime in Version 3. A small cache enables local clients to avoid continual, expensive communication with the router for each network message sent.

When a process wants to send a message to an arbitrary node, it invokes *nsend()*, which first consults its own route cache or the router. The obtained 'route' indicates the local 'forwarding process', a process running on the same node that will receive the message and forward it towards its destination. Typically the forwarding process is a datalink output process, the owner of an output link. Whenever a datalink output gets a message it simply pumps the message out of its link, which could be a transputer link, a Unix device driver or a TCP/IP connection. Datalink input processes wait for a Trollius network message to arrive on the same links. It calls *nsend()* on every message it receives and the procedure just described repeats itself. When a message has reached the correct node, the forwarding process is the receiving destination process specified by the original caller of *nsend()*.

Once the forwarding process is known at each stage, *nsend()* actually moves the message to the forwarding process by calling *dsend()*. If a process knows the forwarding process in advance, it need not consult the router and can call *dsend()* directly.

At no point does any process consider dropping a network message. If *dsend()* detects that a forwarding process is busy, it must either wait or pick another forwarding process. If a datalink input must wait, no other message can be read from its link. A clogged link can be a bottleneck and can cut off control over a node by the user seated at the host node. A need exists for message buffers.

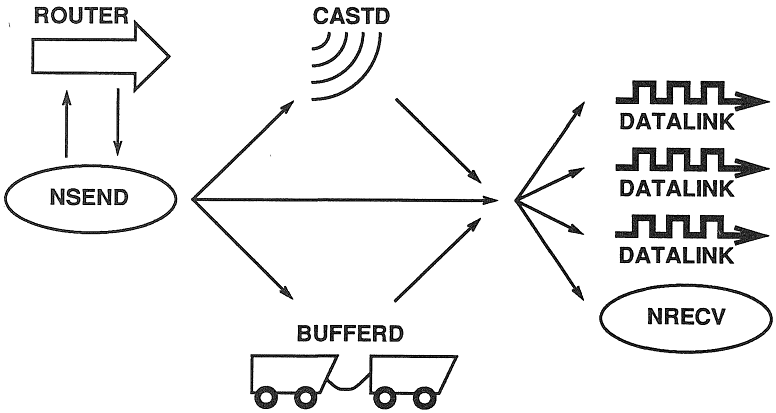


Figure 11: Network Forwarding Processes – After consulting the router, a process calling *nsend()* transfers the message to a local forwarding process. *Bufferd* can hold multiple messages, up to a space limit tuned by the user.

Bufferd is a network layer forwarding process that manages a pool of message buffers and a small group of worker processes. Unless forbidden by its caller, *dsend()* can forward its message to *bufferd*. The buffer daemon forwards the message to an idle buffer worker which then blocks until the original forwarding process is free. If no worker is available, the message waits in a large pool inside *bufferd*. When this is exhausted, callers to *dsend()* wait on *bufferd*. Users can tune the number of workers and the size of the message pool from the command line.

Another forwarding process is the *cast* daemon. Within the domain of physical node identifiers are cast identifiers which refer to groups of nodes. A few useful casts – like broadcast – are predefined, but all others are explicitly chosen and registered by user processes, which then use them with *nsend()* as though they were normal destination nodes. When *nsend()* consults the router about a cast identifier, it is directed to forward its message to *castd. Castd* then performs the necessary *nsend()* operations to the correct node neighbours which reflect an efficient spanning tree to the cast members. *Castd* is completely topology independent. *Castd* demonstrates the modularity of the Trollius network layer. Services can be unplugged, modified, and re-inserted with little concern for the rest of the system. Forwarding processes can be implemented for new network services and only the programmable router is any the wiser.

Local layer

The real messages in Trollius are the local or kernel layer messages. A network message comprises two kernel messages; the first contains the network message descriptor and the second contains the actual data. Forwarding a message with *dsend()* really means sending kernel messages to a local process with *ksend()*. Processes synchronize on two arbitrary numbers: a message event and a message type. Synchronizing events must equal in value and the types must match in at least one bit (bitwise AND is non-zero) for a message to be transferred. The 'indication of forwarding process' alluded to earlier is the correct event and type. Local message passing calls block a process until it synchronizes unless the associated *ktry_send()* polling calls are used.

Synchronization is managed by the kernel, which in Trollius is a server not an executive. The kernel is considered a server because it is not in complete control of the processor. Final scheduling and ultimate process creation/destruction is handled by the underlying environment: the host node's native operating system or the transputer's microcode. On other compute node processors, these responsibilities are handled by a software sub-kernel. Further explanation of the sub-kernel is left to another publication.

Although the rendezvous between two processes occurs through the kernel, actual data is transferred directly, without the kernel's aid. The rendezvous tells each process what it needs to know in the underlying environment to communicate directly with its peer (on the transputer, this would be a channel address). Directed by a flag, the *ksend()* runtime function can remember the results of synchronizing on multiple events and then bypass the kernel the next time a known event is indicated. The capability is called a virtual circuit which effectively welds a dynamic group of processes into a static group. When used between callers of *nsend()* and datalink processes, an efficient virtual circuit can exist between any two points in the multicomputer. The lower overhead of virtual circuits makes them the preferred communication mechanism for nearest-neighbour domain-decomposed applications seeking finer granularity. (Note: CrOS has an option for virtual circuits.) A circuit that crosses a link reserves that link for the caller of *dsend()*. This facilitates the lowest level call in the network layer, *psend()*, which drives the hardware directly.

The predominant responsibility of the spartan local kernel is message passing among dynamic local node processes. It does not know of multiple processors.

Since many underlying environments do not supply a fully developed, predictable priority-scheduling service, the Trollius kernel has an option to schedule processes – by controlling replies to kernel requests – to the the underlying environment. When a client process attaches to the kernel it can decide to be uncontrolled, in which case scheduling is completely handled below. Alternatively the client process can supply a priority, in which case Trollius guarantees that only the highest priority unblocked process runs.

Another kernel feature is asynchronous signal delivery, modelled on Unix. Signals may only be caught during kernel requests because the kernel is a server. Clients can detach from the kernel and go about their business independently. The kernel's server design is the key to Trollius uniformity and portability between host and compute nodes.

Toolkit layer

The toolkit layer is not formally part of the operating system because it does not deal in Trollius messages. Instead, the tools concentrate on booting the host nodes and device connected compute nodes. It would be possible to burn Trollius into PROMs on compute nodes, but again a modular approach is followed. As with all other layers the philosophy guiding the toolkit layer is to support other solutions that could fit into the next ascending layer, not just the one chosen by Trollius. Since the toolkit layer boots Trollius onto certain nodes, the same tools could presumably be used to boot anything.

A configuration file called a process schema indicates the programs that constitute the operating system on each type of node. The *hboot* tool uses one process schema to start processes on a host node. The *sboot* tool uses another process schema to start processes on a device connected compute node. *Sboot()* invokes the *patch* tool to relocate addresses (given a compute node with no dynamic address translation) and then the *cboot* tool to actually load a program. *Cboot* communicates with the Trollius bootstrap loader, called *moses*, running on the device connected compute node. *Moses* is loaded by the export tool, which talks to whatever boot mechanism exists after reset. *Moses* can also be burned into PROM. The reset is achieved with the *fish* tool, the simplest, lowest order of functionality in Trollius and a very long way from the *spread* tool.

How fast will it go?

The following performance numbers were obtained on a T800 (20 MHz) using the LSTT C compiler under the latest version of Trollius (2.0 He). Both code and data resided in external memory. The measurements made between two nodes involved directly connected neighbours, and no C004 or other link reconfiguration switch.

The time per message per hop T is

$$T = T_s + (T_c \times L) + (T_p \times E_p)$$

where

T is time per message per hop (μs),
T_s is set up charge (μs),

$$\begin{aligned} T_c &\ \text{is}\ \ \text{communication time } (\mu\text{s/byte}), \\ T_p &\ \text{is}\ \ \text{packetization cost } (\mu\text{s/extra-packet}), \\ L &\ \text{is}\ \ \text{message length (bytes), and} \\ E_p &\ \text{is}\ \ \text{number of extra packets (after the first one)} \end{aligned}$$

and

$$\begin{aligned} E_p &= \text{ceil}(L/L_p) - 1, \text{ for } L > 0 \\ &= 0, \text{ for } L = 0 \end{aligned}$$

where

$$\begin{aligned} \text{ceil}() &\ \text{is}\ \ \text{the integer ceiling function} \\ L_p &\ \text{is}\ \ \text{the packet size (bytes) (4096).} \end{aligned}$$

Network messages are limited in size. Larger messages are packetized. The default packet size is 4096 bytes although this can be changed before compiling Trollius.

A high overhead is incurred when a message passes through a buffer. Buffers can prevent deadlock and reduce blocking. In practice, they tend to enhance the robustness of the system. Trollius does not choke if a process accidently sends 1000 messages to a node with no receiving process, but at a substantial cost. Message buffering can be disabled by setting the *NOBUF* flag in the *nh_flags* field of the message header. The cost per message buffered is

$$T_b = 2375 + (0.20 \times L) \text{ (in } \mu\text{s).}$$

At each communication layer, regular and virtual circuit message passing have been measured between two directed connected nodes ($n_0 - n_1$) and within one node (n_0). To help in comparing and deciding which of the communication layers is most suited for a particular application, a quick summary of the functionality provided by each layer is given below.

Nsend: network layer
- *routed:* from/to any node
- *multitask:* from/to any node process
- *packetized:* any message size

Dsend: datalink layer
- *not-routed:* nearest neighbour only
- *multitask:* from/to any node process
- *not-packetized:* message size limited to L_p (default: 4096)

Ksend: kernel 'layer'
- *not-routed:* local communication only
- *multitask:* from/to any process on the same node
- *not-packetized:* no limits on message size but with protection against mismatched lengths

Psend: physical layer
- *not-routed:* nearest neighbour only
- *not-multitask:* single process communication
- *not-packetized:* no limits on message size but no protection against mismatched lengths can only be used after a virtual circuit has been set up

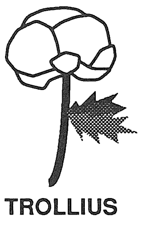
Regular			Virtual Circuit			
T_s	T_c	T_p	T_s	T_c	T_p	
480	.56	552	168	.56	140	Nsend (n_0–n_1)
388	.10	374	211	.10	195	Nsend (n_0)
453	.56	—	128	.56	—	Dsend (n_0–n_1)
355	.10	—	177	.10	—	Dsend (n_0)
230	.10	—	69	.10	—	Ksend (n_0)
—	—	—	36	.56	—	Psend (n_0–n_1)

During the development and debugging of an application program, use of *nsend()* is recommended. When the program is working, *nsend()* can be substituted for *dsend()*, *ksend()* or *psend()* where possible. Although this provides enough flexibility for most users, it is possible that none of the above suits a particular application. For such requirements, the project has developed the Custom Network Gearbox approach to tailor the communication functionality/performance to applications that fall into the cracks of the regular Trollius communication layers. This approach is explained in a paper published in the proceedings of the Fifth Distributed Memory Computing Conference.

State of the project

The first version of Trollius was written at the Cornell Theory Center. The recent second generation Trollius (described here) and the upcoming third generation is based at The Ohio State University, Research Computing. The project also continues at Cornell with Dave Fielding, Moshe Braner, Jim Beers and Roslyn Leibensperger. Source level releases for academic sites are available from OSU for $300 US. Industrial sites are encouraged to make a larger donation. Two compiler options, the Logical Systems Transputer Toolset or Genesys are licensed separately and bundled with Trollius. The centre of Trollius expertise in Europe is at K-par Systems Ltd, Bristol in the UK. An electronic mail discussion group is operated from OSU. To participate or to make other inquiries or documentation requests, write or e-mail to

Trollius Project   trollius@tbag.osc.edu
Research Computing
The Ohio State University
1224 Kinnear Rd.
Columbus, OH 43212
United States of America



The goal of Trollius would be to create or influence a standard that would increase productivity and generate enough comfortable software to capture the more conservative supercomputer user. If accomplished, the objective would sell hardware, make money, and facilitate more science.

A TRANSPUTER SIMULATOR

S. Bakhteyarov, E. Dudnikov, M. Yevseyev and A. Semion
International Research Institute for Management Sciences, Moscow, USSR

A program is described which simulates the INMOS T414 transputer at the memory transfer and instruction set level. The program is implemented on the IBM PC XT/AT. It shows most of the changes in the internal state of the transputer while running a transputer step-by-step. The program can be a useful tool for debugging transputer assembly programs.

Introduction

In recent years the sophistication of parallel computer architectures called for development of transputers, a new kind of microprocessor. Since starting in 1983, INMOS is now manufacturng a third generation of transputers [18]. Transputers are widely used as PC accelerators, in various dedicated data processing systems, in control systems, and as main processors of new generations of workstations and desktop supercomputers.

With this broad spectrum of applications the importance of debugging transputer programs increases dramatically [19]. The debugging, i.e. fault detection, localisation and correction procedure may be implemented for a high-level programming language (occam for example) or for an assembler or even the transputer machine code. In recent years a growing interest has been shown in development of comprehensive hierarchical debugging systems for users needing to debug occam programs, but which can whenever necessary descend to a lower level and examine the behaviour of some code fragments on the transputer instruction set level.

This paper provides a brief description of one such debugging tool called the *Transputer Instruction Set Simulator* (TISS) developed in the International Research Institute for Management Sciences, Moscow, USSR.

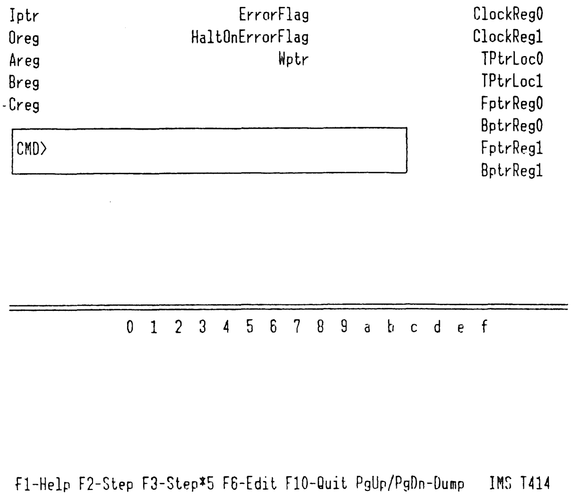
TISS simulates the INMOS T414, the same approach has also been used to implement an instruction set simulator of the INMOS T800. The simulator may be used not only as a debugging tool, but also as a teaching aid for studying the transputer architecture and instruction set.

The TISS works on IBM PC XT/AT compatible machines under MS–DOS 3.3 with a RAM of at least half a megabyte.

Purpose and capabilities

TISS is a software model of the INMOS T414 transputer implemented on the instruction set level. A loadable T414 program file (a '.b4' file) is used as the input data. Such a file can be created by a transputer assembler designed for use with the T414 or T800.

The simulator disassembles the loaded file and displays the mnemonics of the instructions on the screen. The internal state of the transputer (the register stack, the instruction pointer and the workspace pointer) are also displayed on the screen with all registers of the scheduler and those of the timers. The user also can view the memory region occupied by the program, and its workspace region, and can watch



```
Iptr                     ErrorFlag                 ClockReg0
Oreg             HaltOnErrorFlag                   ClockReg1
Areg                          Wptr                  TPtrLoc0
Breg                                                TPtrLoc1
Creg                                                FptrReg0
                                                    BptrReg0
CMD>                                                FptrReg1
                                                    BptrReg1

================================================================
         0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f


F1-Help F2-Step F3-Step*5 F6-Edit F10-Quit PgUp/PgDn-Dump   IMS T414
```

Figure 12: Main simulator screen

changes in the memory produced by the executing program.

The simulator can execute the loaded program step-by-step, set the initial address from which execution begins, edit the code in text and hexadecimal, edit the register contents, or write the edited and debugged program as a disk file.

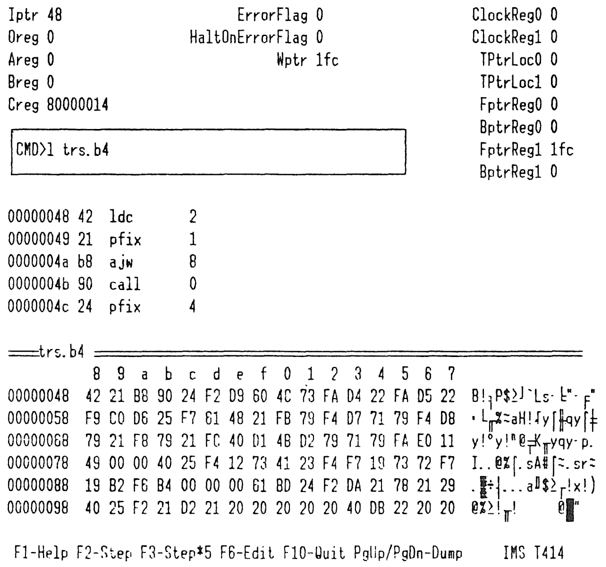
General description

Loading the simulator is done in the usual way in the DOS environment, by typing the name of the program (TISS) at a DOS prompt, and pressing enter. The program identifier is displayed on the screen until a key is pressed; then the main simulator screen is displayed. The main screen contains the transputer registers, the binary and mnemonic representation of the code and the simulator command prompt. This screen is shown in figure 12.

The upper part of the screen shows the transputer registers and simulator command prompt. The lowest line of the screen is a quick-reference line. By pressing *F1* the user may view the help text shown in figure 14, *F2* executes the loaded program step-by-step, *F6* gets the user into the editor, *F10* exits into DOS. *PgUp/PgDn* may be of use when viewing the loaded code.

The simulator commands

The simulator commands are divided into those entered from the keyboard having parameters, and those which have no parameters and are executed by pressing a



```
Iptr 48                     ErrorFlag 0             ClockReg0 0
Oreg 0                HaltOnErrorFlag 0             ClockReg1 0
Areg 0                           Wptr 1fc            TPtrLoc0 0
Breg 0                                               TPtrLoc1 0
Creg 80000014                                        FptrReg0 0
                                                     BptrReg0 0
CMD>l trs.b4                                         FptrReg1 1fc
                                                     BptrReg1 0

00000048 42  ldc      2
00000049 21  pfix     1
0000004a b8  ajw      8
0000004b 90  call     0
0000004c 24  pfix     4

═══trs.b4 ══════════════════════════════════════════════════════════
          8  9  a  b  c  d  e  f  0  1  2  3  4  5  6  7
00000048  42 21 B8 90 24 F2 D9 60 4C 73 FA D4 22 FA D5 22  B!╕P$≥┘`Ls·╘"·╒"
00000058  F9 C0 D6 25 F7 61 48 21 FB 79 F4 D7 71 79 F4 D8  ∙└╓%≈aH!√y⌠╫qy⌠╪
00000068  79 21 F8 79 21 FC 40 D1 4B D2 79 71 79 FA E0 11  y!°y!ⁿ@╤K╥yqy·p.
00000078  49 00 00 40 25 F4 12 73 41 23 F4 F7 19 73 72 F7  I..@%⌠.sA#⌠≈.sr≈
00000088  19 B2 F6 B4 00 00 00 61 BD 24 F2 DA 21 78 21 29  .▓÷┤...a╜$≥┌!x!)
00000098  40 25 F2 21 D2 21 20 20 20 20 20 40 DB 22 20 20  @%≥!╥!     @█"

 F1-Help F2-Step F3-Step*5 F6-Edit F10-Quit PgUp/PgDn-Dump     IMS T414
```

Figure 13: The simulator screen after loading *trs.b4*

function key. A description of the parametric (keyboard) commands is displayed by pressing *F1*, a description of non-parametric ones is displayed on the quick-reference line.

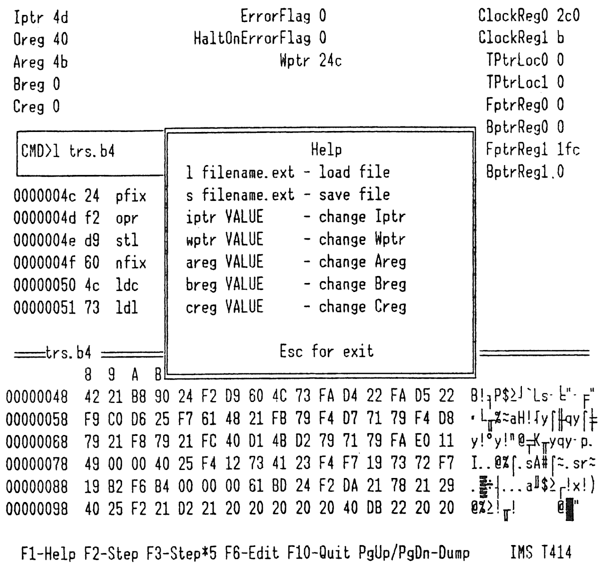
There are seven keyboard commands. They are file-loading and -saving commands, and commands for setting the values of registers. They are:

l *file_name*	load file
s *file_name*	save file
iptr *value*	set **Iptr** to value
wptr *value*	set **Wptr** to value
areg *value*	set **Areg** to value
breg *value*	set **Breg** to value
creg *value*	set **Creg** to value

After loading a file, for example *trs.b4*, the screen will look like that in figure 13. Once the file is loaded **Iptr** is set to **48h** (the address of the start of the program), **Wptr** is set to the address of the first free word in the memory above the loaded program (in this case `1FC`), **Creg** is set to the address of the program load channel (*Link1in*, which is `0x14`).

Below the register area five instructions of the disassembled code are shown, and below that is the dump of the first sixty bytes of the code in hex and symbol format with byte addresses.

The state of the screen after a few steps of the program is shown in figure 15. The contents of the registers have been changed. Also *F6* was pressed so the quick-reference line is modified.



```
Iptr 4d                  ErrorFlag 0             ClockReg0 2c0
Oreg 40            HaltOnErrorFlag 0             ClockReg1 b
Areg 4b                       Wptr 24c            TPtrLoc0 0
Breg 0                                            TPtrLoc1 0
Creg 0                                             FptrReg0 0
                                                   BptrReg0 0
CMD>l trs.b4          Help                         FptrReg1 1fc
             1 filename.ext - load file            BptrReg1.0
0000004c 24  pfix   s filename.ext - save file
0000004d f2  opr    iptr VALUE     - change Iptr
0000004e d9  stl    wptr VALUE     - change Wptr
0000004f 60  nfix   areg VALUE     - change Areg
00000050 4c  ldc    breg VALUE     - change Breg
00000051 73  ldl    creg VALUE     - change Creg

═══trs.b4 ═══            Esc for exit              ═══════
          8  9  A  B
00000048  42 21 B8 90 24 F2 D9 60 4C 73 FA D4 22 FA D5 22  B!┐P$≥┘`Ls·╘"·╒"
00000058  F9 C0 D6 25 F7 61 48 21 FB 79 F4 D7 71 79 F4 D8  ·└╓%≈aH!√y⌠╫qy⌠╪
00000068  79 21 F8 79 21 FC 40 D1 4B D2 79 71 79 FA E0 11  y!°y!ⁿ@╤K╥yqy·α◄
00000078  49 00 00 40 25 F4 12 73 41 23 F4 F7 19 73 72 F7  I..@%⌠.sA#⌠≈.sr≈
00000088  19 B2 F6 B4 00 00 00 61 BD 24 F2 DA 21 78 21 29  .▓÷┤...a╜$≥┌!x!)
00000098  40 25 F2 21 D2 21 20 20 20 20 20 40 DB 22 20 20  @%≥!╥!     @█"

 F1-Help F2-Step F3-Step*5 F6-Edit F10-Quit PgUp/PgDn-Dump    IMS T414
```

Figure 14: Screen with help text

Loading and saving a file

To load a file enter the command

```
CMD>  l file_name
```

When loading is complete the user may view the file by pressing the *PgUp/PgDn* keys.

To save the file enter the command

```
CMD>  s file_name
```

File editing

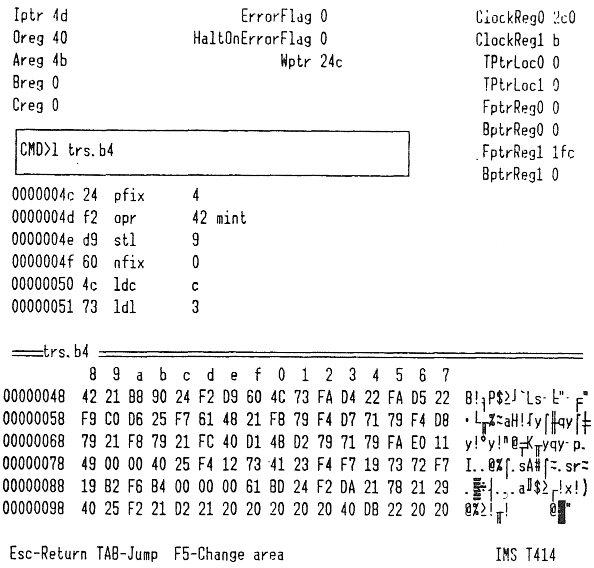
To enter edit mode press the *F6* key. The cursor jumps into the address area. Here the user may enter an address, and the code resident at that address will be displayed. To change hex/symbol area press the *TAB* key, and to change hex/symbol area press *F5*. To leave edit mode press *Esc*.

Stepping the program

To step through the program press the *F2* key. The instruction pointed at by Iptr will be executed. This instruction is pointed at by the arrow on the screen. The last instruction executed is saved on the screen.

To start execution from any address, for example 120, set Iptr to that address by the command

```
CMD>  iptr 120
```



```
Iptr 4d                  ErrorFlag 0          ClockReg0 2c0
Oreg 40          HaltOnErrorFlag 0            ClockReg1 b
Areg 4b                       Wptr 24c         TPtrLoc0 0
Breg 0                                         TPtrLoc1 0
Creg 0                                         FptrReg0 0
                                               BptrReg0 0
CMD>l trs.b4                                   FptrReg1 1fc
                                               BptrReg1 0
0000004c 24  pfix      4
0000004d f2  opr       42 mint
0000004e d9  stl       9
0000004f 60  nfix      0
00000050 4c  ldc       c
00000051 73  ldl       3

══trs.b4 ═══════════════════════════════════════════════════
          8  9  a  b  c  d  e  f  0  1  2  3  4  5  6  7
00000048  42 21 B8 90 24 F2 D9 60 4C 73 FA D4 22 FA D5 22  B!╕P$≥┘`Ls·╘"·╒"
00000058  F9 C0 D6 25 F7 61 48 21 FB 79 F4 D7 71 79 F4 D8  ·└╓%≈aH!√y⌠╫qy⌠╪
00000068  79 21 F8 79 21 FC 40 D1 4B D2 79 71 79 FA E0 11  y!°y!ⁿ@╤K╥yqy·α.
00000078  49 00 00 40 25 F4 12 73 41 23 F4 F7 19 73 72 F7  I..@%⌠.sA#⌠≈.sr≈
00000088  19 B2 F6 B4 00 00 00 61 BD 24 F2 DA 21 78 21 29  .▓÷┤...a╜$≥┌!x!)
00000098  40 25 F2 21 D2 21 20 20 20 20 20 40 DB 22 20 20  @%≥!╥!     @█"

Esc-Return TAB-Jump  F5-Change area                 IMS T414
```

Figure 15: Screen state following execution of several steps, and with edit mode on

Summary

A transputer simulation at the instruction set level has been described. The simulator was designed for debugging transputer programs and may be used either as a stand-alone debugging tool or as a part of a complex debugging system. In either case the simulator would be used by a professional programmer already acquainted with the transputer instruction set and architecture. However the simulator may also be used as a teaching guide for those studying the structure of the transputer and its functioning.

References

[18] INMOS Ltd, *Transputer reference manual*, Prentice Hall, 1988.

[19] C. O'Neil, *The TDS occam 2 debugging system*, in Parallel programming of transputer based machines, IOS, 1988.

[20] INMOS Ltd, *The transputer instruction set: a compiler writer's guide*, Prentice Hall, 1988.

Evgeny E. Dudnikov
International Research Institute
for Management Sciences
9, Prospekt 60–let Oktyabria
117312 Moscow
USSR

Tel: +7 095 135-52-58
Fax: +7 095 135-12-34

PRODUCTS, SERVICES AND ANNOUNCEMENTS

OCCAM PROMOTION INITIATIVE

Michael Poole, INMOS Limited

My article in the previous newsletter generated a lot of interest, in spite of the fact that at the time of publication my electronic mail address given at the bottom of the article did not work. I believe that that address does now work as do the fuller versions given below.

Now that the new TDS3 (IMS D700E) is out, I am able to devote most of my time to this work now.

Source of an occam compiler

The most interest was shown in the intended availability of the source of the new INMOS compiler to people who want to do a retargetting exercise. Unfortunately, for a variety of reasons, this source does not yet exist in a form that would be suitable for such a distribution, and so I must apologise to those of you who had hoped for an off the shelf compiler porting package like the old occam 1 portakit.

What we have at present is a compiler written in ANSI C that compiles occam source to transputer binary in TCOFF (Transputer Common Object File Format) files. This compiles the complete occam 2 language and includes all the compile time checking for which occam has been designed. For a successful retargetting exercise the missing component is an adequate description of the intermediate representation of the program being compiled, so that alternative output code for another architecture could be designed and a suitably modified compiler written. The transputer back end of the compiler is still undergoing pre-release evaluation, and not until this has got to product will it be sensible to make the source into a distributable package.

Meanwhile I do urge readers who have a definite intention to perform a retargetting exercise, and especially those who hope to derive a distributable product, firstly to sign the necessary IMS D702A licensing agreement with an INMOS sales office, and secondly to write to me informally discussing their intentions. In particular it may be possible for me to introduce people with similar intentions to each other so that the possibility of collaborative projects could be investigated. I should also be interested in hearing from anyone who has funds to invest in this kind of work and who would like an introduction to possible technical partners.

Development of occam

Other activities under way include the extension of occam 2 to include record types and other frequently requested minimal extensions, and work towards a possible international standard. Geoff Barrett of INMOS has submitted a paper to the next OUG technical meeting at York on his proposals for language extensions. Michael Goldsmith and colleagues of Formal Systems (Europe) Limited of Oxford, among other occam related activities, many supported by ESPRIT as part of the PUMA project, are working towards the development of a standard entitled 'occam and its use in verified systems' in the form required by international standards bodies. Such a standard would emphasise the formal syntactic and semantic definition of the language, and would define criteria for demonstration of the conformance of an implementation to such a standard. It would emphasise the appropriateness of the language as a vehicle for the formal description of hardware and software systems as well as its use as a programming language.

I myself am currently working on a revision of the configuration language used to describe multi-processor occam programs and their mapping on to transputer networks. The principal features of this revision will be the removal of explicit channel placement directives from the software description and provision for the convenient description of nearly regular networks, such as rectangular arrays where a few of the processors have additional external connections, etc.

Publication support

Finally I should like to remind you that INMOS have a scheme whereby you can get administrative, and even financial, assistance from us if you write an article for the technical press on any subject showing occam or the transputer in a good light. If this inducement appeals to you, please approach Francesca Pick, INMOS external relations manager, at the same address as myself, before you approach a possible publisher.

Michael Poole
INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
United Kingdom

Tel: +44 454 616616
Email (USA): mdp@isnet.inmos.com
Email (Europe): mdp@uk.co.inmos.isnet

TRANSPUTER EDUCATION KIT

Computer Systems Architects, Provo, Utah

For the first time ever, affordable and understandable multi-processing for personal and classroom use (subsidized by Inmos Ltd and CSA in support of the educational marketplace).

- ▷ PC-compatible add-in board
- ▷ Inmos T400 20 MHz transputer
  - o 10 MIPS 32-bit architecture
  - o 2 kbytes of on-chip 50 ns DRAM
  - o two 20 Mbits/s bidirectional serial links with DMA
  - o hardware process scheduler
  - o two on-chip timers
  - o submicrosecond ineterrupt latency
- ▷ Sockets for 1 or 4 Mbyte DRAM

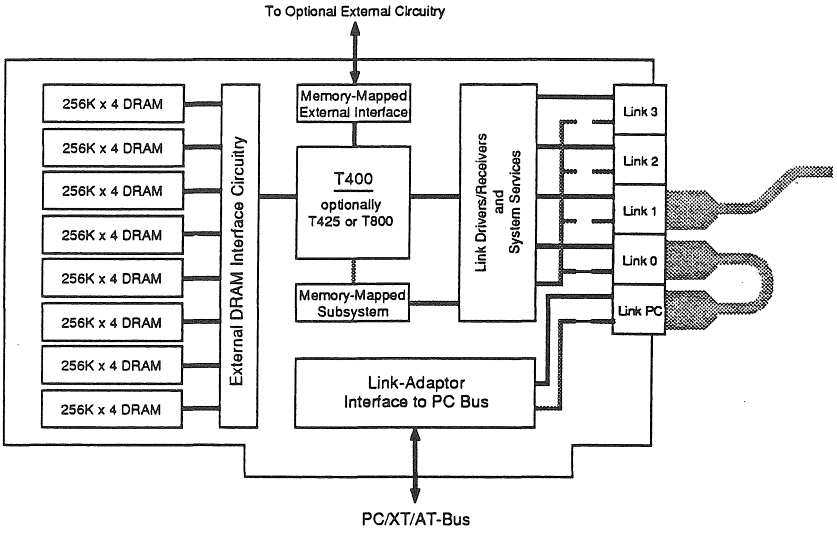

To Optional External Circuitry
256K x 4 DRAM
External DRAM Interface Circuitry
Memory-Mapped External Interface
T400
optionally T425 or T800
Memory-Mapped Subsystem
Link Drivers/Receivers and System Services
Link 3
Link 2
Link 1
Link 0
Link PC
Link-Adaptor Interface to PC Bus
PC/XT/AT-Bus

Figure 16: transputer education kit basic hardware block diagram

- ▷ Designed for operational incorporation within a multi-transputer network
  - o five 8 pin mini-DIN external connectors
  - o accepts industry-standard cables
  - o reliable interconnection at up to forty feet
- ▷ Optional interface for hardware experiments
- ▷ Includes T400 C cross compiler and assembler
- ▷ Includes T400 occam 2 compiler and debugger (requires 1 Mbyte of user-supplied DRAM)
- ▷ Example and demonstration programs
- ▷ More than 1 000 pages of comprehensive documentation (including schematics)
  - o Inmos transputer guide
  - o Transputer education kit manual
  - o Transputer education kit workbook
  - o Logical Systems C toolset manual
  - o Inmos occam toolset manual

Computer System Architects
950 North University Avenue
Provo, Utah 84604-3422
United States of America

+1 (801) 374-2300

THE QUEEN'S AWARD FOR TECHNOLOGICAL ACHIEVEMENT 1990

Oxford University Computing Laboratory

Amongst this year's Queen's Awards were two conferred on OUCL and Inmos recognising their co-operation in applying formal methods in the development of occam and the transputer. This is the first time that a department of the University of Oxford has achieved the distinction of a Queen's Award, and the Computing Laboratory is one of only three university research groups amongst this year's award-winners. The announcement:

> "This award goes jointly to Oxford University Computing Laboratory and Inmos Ltd, for the development of formal methods in the specification and design of microprocessors.
>
> "The occam programming language, developed by the applicants, has been used to translate the IEEE Standard 754 to cover the floating point arithmetic unit for the IMS T800 floating point transputer.
>
> "The use of formal methods has enabled the development time to be reduced by 12 months."

cites both the development of occam, and the particular achievement of getting the T800 transputer FPU *right.*

Occam is the first commercially available language to take seriously the problem of programming MIMD parallel machines. A distinguishing feature is that it goes out of its way to be clear and simple, being designed to be very little more than an implementation of Tony Hoare's CSP. Various work done at OUCL, in particular on a transformational semantics for occam [21], supports reasoning about programs written in the language.

The intellectual leverage gained from the simplicity and elegance of the semantics of occam is instrumental in the success of the work on the transputer design. As well as being used to program transputer-based machines, occam was used in the specification of the components of the microprocessor itself. Perhaps the most tangible commercial success is the design of the floating point arithmetic unit in the T800 transputer.

The two principal actors in this tale are Geoff Barrett and David Shepherd, both of whom came to Oxford as students on the MSc in Computation, and both of whom now work for Inmos Ltd in Bristol. At the time the work was done Geoff was in Oxford doing his doctoral research in the Programming Research Group, and David was designing chips for Inmos.

When the T800 transputer was designed it was decided that in parallel with an informal development, supported by months of testing against other FPUs, a team would set about a formal development of a correct-by-design unit. In this context 'correct' means conforming to the IEEE Standard 754, and a big first step was to decide exactly what the specification required. The IEEE document is partly reliant on the meaning of natural language requirements; Geoff Barrett's recasting of the

Standard into the Z specification language [22, 23] is unambiguous and serves as the touchstone for the correctness of the T800 FPU.

The microcode of the T800 FPU was documented in a restricted subset of occam. Using the occam transformation system developed in Oxford [24], David Shepherd demonstrated that this code was equivalent to a more naturally expressed occam program which had been shown to meet the Z specification. The transformation system builds on the simple semantics of occam, allowing essentially mechanical proof that two programs – like the specification of the FPU and the implementing microcode – have the same effect.

You could hardly have made a better case for formal methods, because the race went to the formal development method. Moreover a number of errors were found in the proposed implementation that had not shown up in months of necessarily very limited testing. (Some of the discrepancies found in testing transpired to be mistakes in an implementation by a competitor, against which the simulation of the T800 was being compared.) Not only is it possible to be much more confident in the accuracy of the finished FPU, but this higher quality implementation was cheaper to design, and completed an estimated year ahead of what would otherwise have been achievable.

The moral of this tale is that formal methods can not only improve quality, but also the timeliness and cost of producing state-of-the-art products. Inmos can be confident that they got it right, and did so on time. The Laboratory has also benefited from the opportunity to develop its work on the underlying theory into something which it has confidence will be useful. There is a very readable non-technical summary of this work in an article in New Scientist last year [25].

The other people in the Laboratory working for Bill Roscoe on the occam transformation project (of which the T800 FPU work was a part) and who implemented the transformation system were Michael Goldsmith and Tony Cox. With Bill and others they are involved in a company called Formal Systems, selling just this kind of expertise, which works out of offices in Oxford and in Auburn, Alabama; and Bill is a lecturer at the University of Oxford.

The work in Oxford on the transformation system and the T800 was supported by Alvey and ESPRIT contracts and by a development contract from the SERC/DTI Transputer Initiative, which the Laboratory gratefully acknowledges. *gj*

References

[21] A. W. Roscoe and C. A. R. Hoare, *The laws of occam programming*, Programming Research Group technical monograph PRG–53, Oxford University Computing Laboratory, February 1986.

[22] Geoff Barrett, *Formal methods applied to a floating point number system*, Programming Research Group technical monograph PRG–58, Oxford University Computing Laboratory, January 1987.

[23] Geoff Barrett, *Formal methods applied to a floating point number system*, in IEEE Trans Soft Eng, May 1989. pp 611–621

[24] Michael Goldsmith, *The Oxford occam transformation system* (documentation), Programming Research Group, Oxford University Computing Laboratory, 1988.

[25] David Shepherd and Greg Wilson, *Making chips that work*, New Scientist, Nº 1664, 13 May 1989. pp 61–64

TRANS-RTXC – THE FIRST TRUE REAL-TIME KERNEL FOR TRANSPUTERS

Intelligent Systems International, Leuven

Now you can match the scalable processing power of the INMOS Transputer with the software power of a multitasking, event-driven real-time kernel. As the only microprocessor specially designed for parallel processing, the Transputer can operate as a single processor or as a node in a distributed network. From a single processor to over a hundred processors, you can achieve raw computing speeds from 10 Mips to over 1000 Mips. To harness all this power for real-time applications, you will need to be able to switch processes dynamically, allocate system resources, and synchronise the operation of tasks, remotely or locally, and do so within tens of microseconds.

You can do all of that, right now. TRANS-RTXc is here, the only real-time kernel specifically designed for the Transputer. It is a special port of the proven real-time kernel RTXC combined with extensions to capture the power of the Transputer. TRANS-RTXc, like RTXC, gives you a task-based organisation with preemptive scheduling, multiple priorities, interrupt management, resource management, inter-task management and synchronisation. ISI has developed TRANS-RTXc in cooperation with the creators of RTXC, A. T. Barrett & Associates. TRANS RTXc is fast. A basic context switch takes less than 6 μs on a 25 MHz Transputer, which is faster than on most other processors.

Real-time performance to be expected from TRANS-RTXc

To indicate the real-time performance you can expect from TRANS-RTXc, we have taken a snapshot of the demo program supplied with the licence. The figures in figure 17 were obtained on a 25 MHz Transputer.

On the native Transputer, the actual response time might be within a few microseconds or, in the worst case, take an additional time equal to $(2n - 2)T_s + T_{sch}$, where n is the number of processors, T_s is the timeslice (1 ms) and T_{sch} is the time to the next descheduling point. As a result, on a native transputer you only know the lower time limit, not the upper limit of the scheduling interval. With TRANS-RTXc, the lower limit is increased by a few microseconds, but your upper limit is known and is around 140 μs. This is *fast pre-emptive scheduling.*

The table in figure 18 summarises these figures (in microseconds). They were obtained on a 25 MHz Transputer. Note that the actual switching time has only increased by 5 μs.

Using standard C code in a single transputer environment, TRANS-RTXc applications are compatible with RTXC application code for 8, 16 and 32-bit architectures. You can move your application to the Transputer with the minimum of change. That

Minimum TRANS-RTXc kernel call	22μs
average allocate and deallocate	33μs
enqueue 1 byte	39μs
dequeue 1 byte	37μs
enqueue 4 bytes	38μs
dequeue 4 bytes	36μs
average lock and unlock resource	29μs
enqueue 1 byte to a waiting higher priority task	167μs
send message to a higher priority task and wait for acknowledgement	179μs
send message to a lower priority task and wait for acknowledgement	153μs
signal/wait plus wait/signal handshake between two tasks	173μs
send 57 byte message over link to lower priority task with fpu swap	192μs
send 57 byte message over link to lower priority task without fpu swap	176μs

Figure 17: Performance figures for TRANS-RTXc services (25 MHz)

	without TRANS-RTXc		with TRANS-RTXc		direct switch
limits	lower	upper	lower	upper	
5 process in queue	1	8100	36	140	6
10 process in queue	1	18100	36	140	6

Figure 18: Limits on response time, in microseconds

saves you money and gets your products to market *fast*. TRANS-RTXc may be used for application development with languages other than C. TRANS-RTXc provides support for Fortran, Pascal, Modula-2 and occam.

With TRANS-RTXc, Intelligent Systems International will take you beyond the single processor Transputer environment and into the world of distributed real-time multi-processing systems. To do so, ISI has extended RTXC with such capabilities as remote task management, remote synchronisation, and remote communication services, each performed with a single call to the TRANS-RTXc kernel. This will enable designers to consider a Transputer network as one single real-time processing unit. The design phase is supported by RTXCgen, a system generation utility and RTXCbug, an integrated debugging tool

For more information, in Europe:

Eric Verhulst	Tel: +32 16-29 01 28
International Systems International	Fax: +32 16-20 80 57
Interleuvenlaan 62	
3030 Leuven	
Belgium	

in North America:

Tom Barrett	Tel: +1 713 728 9688
A. T. Barrett & Associates	Fax: +1 713 728 1049
11501 Chimney Rock, Suite R	
Houston, TX 77035	
United States of America	

OCCAM 2 AND TRANSPUTER ENGINEERING COURSE

Computing Laboratory, University of Kent at Canterbury

Course Objectives To acquire technical knowledge, insight and practical experience of parallel system design using *occam* and *transputer* networks.

Further Details Harnessing the potential processing power of *transputer* networks requires the development of a fluency in parallel systems design equal to our traditional skills for sequential logic. *Occam* is a simple, small but powerful language which enables such fluency. Software engineering principles, load-balancing techniques, real-time applications and various embedded and super-computing issues will be covered. The strengths, weaknesses and likely future developments of *occam* and *transputer* technologies will be discussed.

Course Members If you are thinking of using parallel computing to engineer high-performance high-security systems, this is the course for you. If you have picked up basic *occam* syntax and semantics and are wondering how best to exploit its power, come along. If you have never seen any *occam* before, so much the better! Hardware engineers are especially welcome. *C* programmers beware – this course will change your life!! *[Since September 1986, this course has attracted over 200 participants from Industry and Academia worldwide.]*

Course Methods Informal lectures with a large proportion of 'hands-on' experience being provided through practical exercises and a 'mini-project'. Practical work will be on the MEiKO Computing Surface and will be supervised at the ratio of one tutor for every six attendees. The MEiKO provides a multi-user multi-transputer development and applications environment. Our system will support up to 30 simultaneous users, each with dedicated access to a private network of transputers including at least two T800s. The full system comprises over 131 transputers (including 97 T800s) with a gigabyte distributed file store and three high resolution graphics workstations.

Length & Cost Five days @ £450 (including lunches and light refreshments).

Dates Course Nº 16: 24–28 September 1990.

Contact For a full syllabus, application forms, fees, special arrangements and accommodation, please contact:

Professor P. H. Welch — Tel: +44 227 764000 x7695
Computing Laboratory — Fax: +44 227 762811
The University — Telex: 965449 UKCLIB
Canterbury — Email: phw@uk.ac.ukc
Kent CT2 7NF
England

EEC Recognition This course is one of the foundations for a series of courses and technical workshops entitled '*Training for Transputer Technologies*'. These are being developed under contract with the EEC as part of the Communities Action Programme for Education and Training for Technology (COMETT).

occam is a trade mark of the INMOS Group of Companies;
MEiKO and the Computing Surface are trade marks of Meiko Limited.

DIGITHURST MULTIMEDIA SYSTEM USES ARTIFICIAL INTELLIGENCE

Digithurst Ltd, Royston, United Kingdom

Digithurst's Multimedia system, *Picture Book Professional*, now incorporates links to Artificial Intelligence (AI) routines. These enable data to be prioritised and linked according to sets of rules and matching criteria that can be set up by the user to suit a particular application.

Picture Book Professional, which holds references to live and frozen video, graphics and text, has been designed to present users with concepts and events, rather than just the items and objects in a conventional data base.

Concepts are connected by analogy. There is no way to decide at once, when creating a Picture Book, whether an analogy is good or bad, because to some degree everything is connected to everything else. For example, potato is connected to apple because both are vegetable and round in shape. From apple to snake by Biblical association. From snake to doughnut by formal likeness.

The MicroEye PB Card, the hardware which drives the Picture Book Professional system, has been designed with this in mind. At the heart of the board is an Inmos Transputer to carry out the sorting tasks, which can run in parallel with the host program. The system also supports connections to other transputer-based processing elements such as Digithurst's Hard Card for fast image storage and retrieval, and an Image Processing card. With these additions, several sorting and data processing tasks can run in parallel, for example feature extraction from images and word association from text.

The extent to which a concept or event is linked to another can be expressed as a value from 0 to 100%, based on a rule system which the user can develop for a particular application. An exact replica of the event or concept scores 100%, while one with no detectable match under the rule system applied scores 0%. Picture Book Professional displays similar concepts or events in descending order of match score to the comparison item chosen by the user.

One example might be a catalogue of the religious buildings of Europe. When entered originally into the Picture Book Professional database, they were perhaps linked only through Geographical position, century of construction and religion. However, there may be other links not foreseen by the database creator through which the user may want to link the items of the database, for example though architectural features, or through historic events associated with the buildings. To set up these links, the user runs an automatic matching facility in Picture Book Professional.

Another example would be a database of crimes. Each Picture Book element consists of a selection of pictures, text and video (e.g. security camera recordings) related to the event. When a crime is committed, a new Picture Book element is created by entering the data relating to the new event. When a crime is solved, the database is updated with the information. Although it would not be obvious to an operator entering the data where the similarities or links lie with other crimes, by applying the AI matching routines, a set of links with their relevant scores can be constructed. The filters can be successively tightened to show circumstantial and factual links that have a higher probability of bing true.

This advance in using Artificial Intelligence within a Multimedia system is a logical step in the use of Personal Computers to help with techniques of idea association which are fundamental to the deduction processes inherent in so many human activities. The use of Transputers in Digithurst's range of Multimedia hardware adds a powerful and expandable parallel processing facility, while retaining the overall system inside the familiar and inexpensive PC framework.

For more information contact:

Peter Kruger
Managing Director
Digithurst Ltd
Newark Close
Royston
Herts SG8 5HL
United Kingdom

Tel: +44 763 242955
Fax: +44 763 246313

THE STATIC COLUMN TRAM

Division Ltd

Following the success of their TTG1 and TTG3 graphics TRAM designs DIVISION is proud to announce the *Static Column TRAM* (SCRAM). Designed by DIVISION for the TRANSTECH group of companies, the SCRAM is a size 2 Transputer module, and comprises

- ▷ 25 or 30 MHz T801 Transputer
- ▷ 32 or 128 kbytes of 2-cycle static RAM
- ▷ 1, 2, 4 or 8 Mbytes of fast DRAM

With up to 8 Mbytes of 2-cycle memory at 30 MHz on a size 2 TRAM, the SCRAM conforms fully to the INMOS TRAM specification (including height restrictions), and sets new standards for processor speed and memory density in the highly competitive Transputer industry.

DRAM access in two cycles

The DRAM is driven in static-column mode (more accurately, fast page mode) and achieves two cycle access on page hits, five cycle access on page misses. The open page is 4 kbytes wide, so coherent DRAM accesses will very rarely cause page breaks. note that five cycle access on a T801-30 is still only 150 ns, equivalent to zero wait-state access on a T800-20. Achieving two cycle access on a 30 MHz T801 was a significant engineering exercise, and has been achieved without recourse to custom silicon. The board also contains a refresh generator which operates burst refresh address cycles, significantly reducing the number of page misses. The refresh address presented to the DRAM is highly likely to break the current page, so bursting eight refresh cycles at once enhances the performance by page-breaking eight times less frequently.

Differences between the SCRAM and the INMOS B404

The board is visually very similar to the INMOS B404 TRAM, but achieves significantly higher throughput on the same footprint. This is partly due to the faster clock speed and partly due to its two-cycle rather than three-cycle memory interface. Obviously speed-ups due to memory cycle time are highly application specific, but it should be pointed out that applications can be written with the SCRAM in mind, aiming to maximise coherent DRAM access.

Design rationale

To get the best performance from the SCRAM the program's workspace would reside on-chip, code should be in static RAM, and vector space in DRAM. If this is achieved (and the INMOS IMS D700D occam compiler does this automatically) then direct workspace access happens in a single cycle, instruction fetches from the static RAM do not break the currently open DRAM page, and two-cycle access to DRAM can happen very frequently. Of course block-moves between on-chip RAM and DRAM or between static RAM and DRAM will consistently access the DRAM in two cycles. Block-move bandwidth to off-chip memory is consequently increased to 40 Mbytes/s. Note that initializing a word-vector by clearing location zero and block-moving `SIZE`−1 words from location zero to location one will give two-cycle memory accesses and achieve 30 Mbytes/s.

Benchmark figures

No Whetstones, no Dhrystones. DIVISION has benchmarked some code fragments and applications on the IMS B404 (20 MHz) and on a 25 MHz SCRAM. (30 MHz T801s were not yet available for benchmarking.) The SCRAM was also benchmarked against the INMOS B004, but these results are not presented – the B004, although the most popular Transputer board on the planet, is an outdated and slow design, and direct comparisons would be unfair. We have tried to be as fair as possible – these are pieces of code written under consultancy for clients, or written at DIVISION for demonstration purposes. They are therefore very meaningful results, presented exactly as they fell out of the machine:

Program	*Performance ratio: SCRAM to B404*
3D geometry engine	1.25
4×4 matrix inverter	1.36
Distributed polygon shader	1.36
Anti-aliasing CIG shader	1.52
Database search	1.55
Page description language	1.61

The first two tests are small, highly floating-point intensive code fragments, so memory accesses are not expected to dominate – the result of the first program (the worst relative speed performance) is 5% better than a linear scaling of the processor speeds, a reflection of the fact that almost everything had been carefully squeezed on-chip. The CIG shader is an experimental computer graphics program, and is 'classic' occam, seven parallel processes communicating and exhibiting highly conditional behaviour. The last pair of examples are large sequential pieces of code which have not undergone painstaking on-chip optimisation. The 50% to 60% performance improvements over the IMS B404 appear to be the typical figure for larger programs where manual on-chip tuning cannot be done, or where the memory requirements force workspace off-chip.

Application: scientific programming

The SCRAM will make an excellent FORTRAN accelerator – 8 Mbytes of very fast memory on a size 2 TRAM allows substantial dusty decks to be run unmodified via the 3L compiler, without the investment of manual parallelisation. Similarly for large pieces of C code, which could previously not be run on non-virtual memory systems such as transputers, due to memory restrictions. For those users prepared to parallelise code, five SCRAMs can be fitted onto one PC motherboard, giving excellent scientific program performance: 11 MFLOPs and 40 Mbytes of RAM in one PC expansion slot.

Phil Atkin, Charles Grimsdale — Tel: +44 454 324527
Division Ltd — Fax: +44 454 323059

HELIOS NOW RUNNING STANDALONE

Distributed Software Limited, Bristol, UK

Recent work on the Helios Filing System means that fully standalone transputer systems running Helios can now be built, without the need for a host computer (e.g. PC or Sun). The Helios Filing System supports the Inmos B422 SCSI TRAM and other proprietary systems. Work is also underway to provide ethernet support for Helios; this should be completed later this year.

Helios performance benchmarks

A Helios Technical Report which gives a detailed performance evaluation of Helios has recently been published. It will be of interest to any transputer systems designer who needs quantitative measures of the (relatively low) overheads imposed on a transputer system by an operating system such as Helios.

Copies of the Technical Report (№ 22) are available free of charge from the address below.

Helios will be demonstrated on Stand 10 at the *Transputer Applications 90* exhibition at Southampton, 10–12 July 1990.

DSL — Fax: +44 454 618188
670 Aztec West
Bristol BS12 4SD
United Kingdom

TRANSTECH NEW PRODUCTS

TTM100 Intel i860 TRAM

- ▷ Intel i860 40 MHz 64-bit Microprocessor
- ▷ IMS T805 floating point transputer
- ▷ 5 to 20 Mbytes of fast DRAM
- ▷ Sub-system control of reset, analyse and error
- ▷ Communicates via 4 transputer serial links
- ▷ Industry standard size 6 TRAM format
- ▷ Software drivers and maths library support

The Transtech TTM100 contains an INTEL i860 64-bit microprocessor, an IMS T805 floating point transputer, and 5 to 20 Mbytes of fast DRAM. The T805 is configured with 1 or 4 Mbytes of local memory, and shares the other 4 or 16 Mbytes with the i860.

The interface between the i860 and the shared memory system has been optimised to give the i860 zero wait state access for page coherent memory cycles, such as cache fill and cache flush operations. The i860 busLock function is supported for operating system and other special non-divisible memory cycles.

The inclusion of local RAM for the transputer allows both the i860 and transputer to operate concurrently. A busLock mechanism is included in the transputer shared memory interface to optimise block move operations between local and shared memory.

Synchronisation of the transputer and i860 is achieved by a dual event mechanism. This allows either processor to interrupt the other. In this way, either processor can assume the rôle of system master.

The TTM100 returns the performance of up to twenty-four 30 MHz IMS T805's, using the i860 which is capable of 80 MFLOPS (peak single precision), 60 MFLOPS (peak double precision) and 85K Dhrystones.

The TTM100 is supplied with drivers for use with the Occam TDS, Toolset and 3L compilers, together with an array of over 200 vector library routines optimised to take full advantage of the power of the i860.

TTM32/34/38 high performance T801 TRAMs

- ▷ IMS T801 floating point transputer
- ▷ 2, 4 or 8 Mbytes of fast 2 cycle page mode DRAM
- ▷ 32 kbytes of 2 cycle SRAM
- ▷ Sub-system control of reset, analyse and error
- ▷ Communicates via 4 transputer serial links
- ▷ Industry standard size 2 TRAM format

The Transtech TTM32, 34 and 38 consist of an IMS T801 floating point transputer, 2, 4 or 8 Mbytes of fast 2 cycle page mode DRAM and 32 kbytes of 2 cycle SRAM. They have a sub-system port to control reset, error and analyse.

Fast access to the RAM is obtained by doing a very quick address comparison on the present DRAM row address and the last DRAM row address, if they are the same the RAM is accessed in two cycles, if they differ the RAM is cycled in the conventional way. This is known as a page break.

The TTM32, 34 and 38 also have 32 kbytes of fast SRAM, placed just above internal RAM, which does not cause page breaks when accessed. This allows two areas of very fast external RAM to be available at all times, one of which moves through store with the last DRAM address.

The refresh generator works in burst mode, doing 8 refreshes sequentially so that refreshing, which causes page breaks, has less effect on fast page accesses

The fast page is done on 9 bit address boundaries so a page of 512 32-bit words or 2 kbytes of 2 cycle RAM can be accessed in store at any one time.

TTG-F video framegrabber TRAM

▹ IMS T800 floating point transputer
▹ 1 or 4 Mbytes of fast DRAM
▹ 1 Mbyte VRAM
▹ Real time image capture
▹ Supports up to 1024 × 1024 8-bit pixels
▹ Sub-system control of reset, analyse and error
▹ Communicates via 4 transputer serial links
▹ Industry standard size 4 TRAM format

The Transtech TTG-F consists of an IMS T800, running at 25 MHz, floating point transputer, 1 or 4 Mbytes of fast DRAM and 1 Mbyte of VRAM. It also has a Brooktree BT251 video digitiser and an IMSG178 colour palette chip.

The TTG-F is a real time image capturing device for transputer based systems, capable of digitising images of up to 1024 × 1024 8-bit pixels at up to 18 Msps. The 1 Mbyte of Video RAM can hold up to 4 separate images at any one time (depending on image size). Three video inputs are software switchable as well as the resolution being software selectable. The transfer logic can capture single frames or continuous video.

The TTG-F has software support for direct digitisation of PAL and NTSC monochrome video. For capture of RGB 24-bit video three TTG-F's can be synchronised. The TTG-F also has an IMSG178 colour palette chip allowing it to display the live digitised video in the same format as it was captured. The TTG-F also has an IMS T800 plus 1 or 4 Mbytes of DRAM for program use, which allows image processing tasks to be carried out on the TTG-F without the images having to be transferred to other processing modules. However if images need to be transferred all four transputer links are available, enabling real time video at approx 600 × 400 8-bit pixels to be transferred. Coupled with the Transtech TTG1 or TTG3 graphics TRAM's the TTG-F provides a highly flexible image capture, processing and display system.

The TTG-F is supplied with software drivers for programming the options of sampling rate, display resolution and video input as well as an image processing library callable from the Occam TDS, Occam Toolset or the 3L scientific language compilers.

Transputer educational kit ED-KIT 1

- ▹ Transtech TMB03 four TRAM slot motherboard for PC AT/XT Bus
- ▹ TTM6-8-F with IMS T800-20 and 2 Mbytes zero wait state DRAM
- ▹ TTM3-8-F with IMS T800-20 and 1 Mbyte zero wait state DRAM

Transtech have their Transputer Educational Kit on offer incorporating the new TMB03 low cost TRAM motherboard. The TMB03 can accommodate up to four TRAMs, with up to 10 links available on a standard D-type edge connector. Also provided with the Transputer Educational Kit are two of Transtech's range of TRAMs. The TTM6-8-F has an IMS T800 running at 20 MHz with 2 Mbytes of fast zero wait state DRAM, while the TTM3-8-F has an IMS T800-20 with 1 Mbyte of fast zero wait state DRAM.

The following software packages available: Full Occam TDS, Occam Toolset; 3L C, FORTRAN or Pascal

Transtech Parallel Systems Corp.
120, Langmuir Laboratory
95, Brown Road
Cornell Business and Technology Park
Ithaca, NY 14850
United States of America

Tel: +1 607 257 6502
Fax: +1 607 257 3980

Transtech Devices Limited
Unit 17, Wye Industrial Estate
London Road
High Wycombe
Buckinghamshire HP11 1LH
United Kingdom

Tel: +44 494 464303
Fax: +44 494 463686

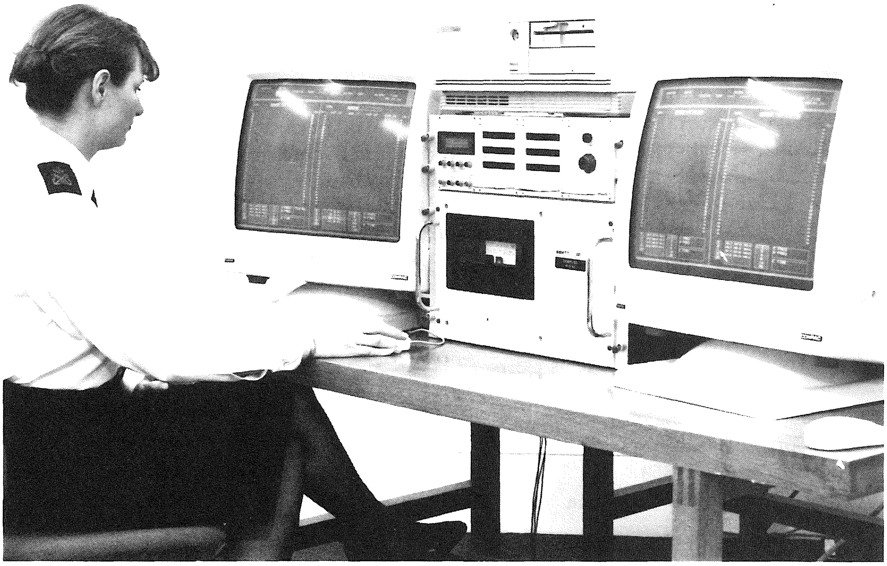
QUINTEK FAST9 ALLOWS PORTABILITY OF NAVAL RESEARCH SYSTEM

Quintek Limited, Bristol

A research and training system for use by the Royal Navy has been developed using the processing power of the Quintek FAST9 transputer board.

There is a requirement to train and test the effectiveness of sonar operators without having to go to sea. Any land simulator has to be extremely accurate so that acoustic traces presented to the operator exactly match those of many real types of vessel, allowing valid conclusions to be drawn on the effectiveness of the complete sonar system.

Department UJP5 at the Admiralty Research Establishment (ARE), Bushey Park, was given the task of developing a system which performed the acquisition and processing of the sonar traces to provide visual output to a monitor. This would allow an operator to detect a new 'contact', identify a type of vessel and to respond accordingly, enabling operator performance research to be undertaken at the shore establishments.



All operator interactions are recorded. The signal processing and display processing can easily be adapted or modified to examine different processing/display algorithms.

The initial equipment used to generate, process and display the images were, respectively, a Dicarps Wide Frequency Spectrum Recorder, a Hewlett Packard 1000 and a Sigmex ARGS Graphics Computer with four 19″ colour monitors, together valued at some £400 000. Although the system worked well, it soon became apparent that the static site was limited for training purposes and a portable version was required which could be transported to Naval establishments, greatly increasing its usefulness.

City Computing Ltd were asked to research the technologies on the market and test a method which would achieve the same results at less cost and which could more easily be moved when required.

The Quintek FAST9 board proved to be the answer. Containing nine T800 transputers with the option of 4 Mbyte on the master and either one or 4 Mbyte on each of the eight slaves, the FAST9 was used to carry out all the necessary processing, acting as an interface between the Dicarps and the PC. Once this system had been proved, a fully working version was tested which together with the FAST9 included a Real World Graphics card driving a monitor giving 1280 × 1024 pixels at 8 bits and a transputer-based disc card for local storage. Controlled by windowing software written by City Computing for the graphics card, and using a mouse to access the sonar functions, the FAST9 handles 150 kbytes of data per second. The system was adapted to accept data from a submarine version of Dicarps and the FAST9 was fully integrated without any modifications, providing a compact and cost-effective solution.

Commenting on the portable system, Dr Chris Klimpke of City Computing said 'Transputers allowed us to integrate a novel, high-performance system using almost totally off-the-shelf products at a fraction of the static model cost. The parallel power of the FAST9 enabled us to use it as the central processing resource in this unique development.'

The FAST9 is just one of a family of high performance transputer products from Bristol based Quintek, used for high processing throughput and image processing.

Quintek Limited
Southfield House
2 Southfield Road
Westbury-on-Trym
Bristol BS9 3BH
United Kingdom

Tel: +44 272 628196
Fax: +44 272 628717

COMPUTER SYSTEMS – SCIENCE AND ENGINEERING

A quarterly international journal
Butterworth Scientific Ltd

The January 1990 issue of *Computer Systems – science and engineering* was a special issue on the topic of transputer applications:

- ▷ *Editorial*, Professor Ian Wand, University of York, UK.
- ▷ *Top-down approach to the design of occam and transputer-based real-time systems*, C. Ciccarella, G. Patis and G. Valent.
- ▷ *Wisdom: a distributed operating system for transputers*, K. A. Murray and A. J. Wellings.
- ▷ *Concurrent communication and granularity assessment for a transputer-based multiprocessing system*, G. Quingping and Y. Parker.
- ▷ *Multiprocessor with dynamically variable topology*, H. Richter.
- ▷ *Optimal topologies of transputers for different classes of problems*, X-L. Deng, T. Dillon, K. Lew, J. Rankin, E. Smith and D. Suter.
- ▷ *Implementation and performance evaluation of an optimization algorithm on transputers*, A. G. Chalmers, J. W. Hearne, K. J. Cameron and N. E. Ligeti.
- ▷ *Parallel alternation direction implicit method on a network of transputers*, R. K. Cooper and D. A. Peshkin.

The issue also carries a section of recent product announcements and a calendar of conferences, courses, exhibitions and events.

Call for papers

The journal is devoted to the publication of high-quality papers on theoretical developments in computer systems science and their application in computer systems engineering. Original research papers, state-of-the-art reviews and technical notes are invited for publication.

All papers will be refereed, and may be: accepted without change; require amendment and subsequent re-refereeing; or be rejected on grounds of either relevancy or content.

Papers should normally be about 5 000 words plus illustrations, and five copies of a double spaced script typed on one side of A4 paper should be sent to the Executive Editor at:

Butterworth Scientific Ltd	Tel: +44 483 300966
P.O. Box 63	Telex: 859556 SCITEC G
Westbury House	Fax: +44 483 301563
Bury Street	
Guildford	
Surrey GU2 5BH	
United Kingdom	

REFERENCE

The list of new members and additions to the occam and transputer bibliography now appear as separate publications.

CONTACTS FOR RELATED GROUPS

Australian Transputer and Occam User Group

The Australian group will be holding another meeting as this newsletter goes to press, 28–29 June 1990, at the Hotel Lawson, Ultimo in Sydney Australia. The person to contact for details of future activities is:

John Hulskamp	Tel: +61 3 660 2453
Department of Communication	Fax: +61 3 662 1060
and Electrical Engineering	rcojh@oz.rmit.xx.minyos
Royal Melbourne Institute of Technology	
GPO Box 2476V	
Melbourne 3001	
Australia	

French Transputer Users Working Group

Traian Muntean	traian@fr.imag.imag
IMAG-LGI	
b.p. 68	
38402 St Martin d'Heres CEDEX	
France	

Deutschen Occam-Interessengemeinschaft der Transputeranwender

DO IT can be contacted through its secretary:

Heinz Ebert
Im Heidigen 3
5206 Neunkirchen-Seelscheid 2
West Germany

The president is:

Joachim Stender
c/o Brainware GmbH
Gustav-Meyer-Allee 25
1000 Berlin 65
West Germany

and vice presidents are:

Frank Heinemann
c/o Fraunhofer-Institute
Kleiststraße 23–26
1000 Berlin 30
West Germany

Peter Eckelmann
c/o Inmos GmbH
Danziger Straße 2
8057 Eching b. München
West Germany
+49 89 319 10 28

Occam User Group Japan

Contact the Secretary:

Mr Kazuto Matsui
Technical Marketing, INMOS Division
SGS-Thomson Microelectronics K.K.
4F Nisseki-Takanawa Building 2-18-10
Takanawa Minato-ku Tokyo 108
Japan
Tel: +81 3 280-4125
Fax: +81 3 280-4131

The chairman is:

Prof. Tosiyasu L. Kunii
Department of Information Science
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo 113
Japan
+81 3 505 2840

Latin American Transputer Users' Group

For further information, contact the Chairman:

Rafael D. Lins
Av. Dr José Rufino 656
Estância
50.781 – Recife – PE
Brazil

Tel: +55 81 251 0713
Fax: +55 81 326 4880

New Zealand Transputer Users' Group

The NZTUG is still only a small organisation. The Chairman in Bob Hogson, Professor of Production Technology at Massey University. Contact the secretary and treasurer:

Dr Ian Graham
Department of Computer Science
University of Waikato
Private Bag
Hamilton, New Zealand

Tel: +64 71562889 x8204
Fax: +64 71384066
CSNet: i.graham@waikato.ac.nz
JANet: i.graham@nz.ac.waikato

Swedish Transputer User Group

The purpose of STUG is to act as an information exchange forum for transputer users in Sweden, and to stimulate discussion concerning related areas such as parallel programming and parallel processor systems. STUG arranges seminars and publishes a newsletter, supported by Gösta Bäckström AB, who represent INMOS in Sweden.

Martin Tørngren
Maskinelement
Kungl. Tekniska Högskolan
100 44 Stockholm
Sweden

Tel: +46-8-790 7849
Fax: +46-8-723 1730
martin@se.kth.damek

North American Transputer Users Group

NATUG have a permanent organization with a committee of about fifteen members, which receives secretarial support from Inmos Colorado Springs. Contacts for this group are: the Chair, Dyke Stiles; the Secretary of the North American Transputer Users Group, care of Mark Hopkins at Inmos Colorado; and the local agent for newlsetter submissions, who is Lyle Bingham. Their addresses appear on page 94.

ELECTRONIC GRAPEVINES

If you are an electronic mail user, you may want to know about two electronic mailing lists, carrying discussions on occam and the transputer. These offer you a mechanism rapidly to distribute information, short papers, programs, problems, even gossip about Inmos, to the sort of people who may be interested. You may even want to read this sort of thing. We even have subscribers from Inmos who can sometimes be goaded into authoritative declarations.

Each list has distribution points both in the UK and the USA. To join try making contact with the appropriate address: for the occam mailing list contact

	`occam-request@uk.ac.oxford.prg`	(in the UK)
or	`occam-request@sutcase.case.syr.edu`	(in the USA);

for the transputer list contact

	`transputer-request@tcgould.tn.cornell.edu`	(in the USA)
or	`transputer-request@uk.ac.oxford.prg`	(in the UK).

Please choose the contact the address that is nearest you, to reduce duplicated traffic across the Atlantic. The transputer list traffic is also available in newsgroup `comp.sys.transputer` on USENET.

A WORD ABOUT NAMES AND NUMBERS

I have tried to be reasonably consistent about addresses and telephone numbers in the newsletter. Human fallibility excepted, the telephone numbers are all given in the international form: so for example a UK caller should replace the +44 of my number by an initial nought, and in the USA you would just drop the +1 from Lyle Bingham's number.

A word may be in order about London telephone numbers: formerly +44 1 (or in the UK, 01) numbers have been divided into +44 71 (in the UK, 071) and +44 81 (in the UK, 081). Mind you, there seem not to be that many London numbers in the *Newsletter*.

Would that electronic mail was as simple! Again I have tried to be reasonably consistent: UK addresses are quoted big-end first, but in other parts of the world `geraint.jones@uk.ac.oxford.prg` for example, would be given little-end first as `geraint.jones@prg.oxford.ac.uk` and in the UK they prefer American addresses like `csa@adam.byu.edu` the other way, in this case as `csa@edu.byu.adam`. If you can tell whether you need to reverse any address from this newsletter, then you are an expert; but if you cannot, I am afraid you will need the help of an expert.

I have been told that if you are at a BITNET site, turning a big-endian address around does not work for all UK addresses, and in particular that it does not work for addresses at `uk.co.inmos`. It ought to be the case that all UK commercial domain addresses are known at Canterbury – `uk.ac.ukc` – so you may be able to render, for example `oug@uk.co.inmos`, as `oug%uk.co.inmos@ukc.ac.uk`. That particular site address, `uk.co.inmos`, ought to be interchageable with `inmos.com`. *gj*

Dear all, we've found the ideal place to indulge in some occam here in Munich — having a great time (hic) wish you were here

occam® user group · enrolment form

·O·U·G·

To enrol a new member in the Occam user group and/or to report a change of address or other details please copy and complete this form and return it to the Occam User Group Secretary, INMOS Ltd, 1000 Aztec West, Almondsbury, Bristol BS12 4SQ, England.

Name (must be an individual not a company)
Address (up to 6 lines, 32 chars each)

New member ☐
Address change ☐
Correction ☐

The Occam User Group Membership list is held on a computer at the INMOS Bristol office. As this is a computer file holding personal information INMOS are obliged to follow the requirements of the Data Protection Act concerning this file. It is therefore necessary to get the written permission of all members for their data to be included in this file.

Please insert name and address in the box above. Please include a postcode if possible.

The additional information requested below may be of use to the OUG committee. Please ensure that you answer the final question and sign the form.

Telephone number:

Electronic mail address:

Please indicate what type of organisation you belong to by ticking one of the following boxes:

Electronics industry	Software industry	Other industry	Academic	Government	Other (describe)
☐	☐	☐	☐	☐	☐

Please give a brief statement of the nature of your interest in occam and the transputer.

The OUG has several special interest groups (SIGs). Please indicate if you are interested in one or more of these subject areas. If you would help to establish or would join a new group please indicate the subject area(s) of interest:

Artificial intelligence . .	☐	Formal aspects	☐
Graphics	☐	Hardware	☐
Learning	☐	Networks	☐
Numerical methods . .	☐	Operating systems . .	☐
Unix	☐	*other*.........................	☐

The mailing list is now becoming a potentially valuable commodity, but we cannot give everyone access without your approval. At present the names and addresses are known by the OUG administration and are provided to INMOS marketing. They are also published in the Newsletter unless a member specifically requests confidentiality. We are also producing a directory of members which could also include telephone and EMAIL numbers and SIG interests if you give permission by ticking appropriate box below.

My name and address may be published in Newsletters/Directory ☐
My answers to all the questions above may also be published ☐
Lists including my name and address may be passed
to third parties offering relevant products or services ☐
I do not mind who sees the information provided here ☐

Signed .... Date ....

NL 13

SPECIAL INTEREST GROUP CHAIRMEN

Artificial intelligence

Joachim Stender
c/o Brainware GmbH
Gustav-Meyer-Allee 25
1000 Berlin 65
West Germany

Environments

Gordon Manson
Department of Computer Science
University of Sheffield
Sheffield S10 2TN
United Kingdom
+44 742 768555 x5580

Education and training

Roger Peel
Department of Electrical Engineering
University of Surrey
Guildford
Surrey GU2 5XH
United Kingdom
+44 483 509284
roger@uk.ac.surrey.ee

Formal methods

Michael Goldsmith
Formal Systems (Europe) Ltd
Unit 7, The S.T.E.P. Centre
Osney Mead
Oxford OX2 0ES
United Kingdom
Tel: +44 865 728460
Fax: +44 865 793165
michael@uk.ac.oxford.prg

Image processing and vision

Hugh Webber
RSRE
St Andrews Road
Great Malvern
Worcs WR14 3PS
United Kingdom
+44 684 894728
hcw@uk.mod.rsre

Graphical program development tools

Mike Roberts
The Centre for Information Engineering
City University
Northampton Square
London EC1V 0HB
United Kingdom
+44 71 253 4399 x3889/3877
m.roberts@uk.ac.city

Hardware

Denis Nicole
University of Southampton
Department of Electronics
and Computer Science
The University
Highfield
Southampton SO9 5NH
United Kingdom
+44 703 787167
dan@uk.ac.soton.ecs

Numerical methods

Derek Paddon
Department of Computer Science
University of Bristol
University Walk
Bristol BS8 1TR
United Kingdom
+44 272 303030 x4336
derek@uk.ac.bristol.compsci

Real time

André Bakkers
Twente University
EL-BSC Dept.
P.O. Box 217
7500 AE Enschede
Netherlands
Tel: +31-53-892794
+31-53-892790
Fax: +31-53-354003
Telex: 44200 thtes
elbscbks@utwente.nl
elbscbks@henut5.bitnet

NATUG STEERING COMMITTEE

INFORMAL OCCAM USER GROUP COMMITTEE

. . . continued from front cover

The 4th Transputer/occam international conference	17
REPORTS	19
North American transputer users group Fall meeting	19
Letter from Bulgaria	20
Postcard from Exeter	21
How many links does it take to screw in a transputer?	23
North American transputer users group Spring meeting	24
The third transputer/occam international conference	26
Fast routing in Japan	30
SPECIAL INTEREST GROUPS	34
Education and training SIG	34
Real-time SIG	35
Formal methods SIG	37
Hardware SIG	38
Environments SIG	40
Graphical program development tools SIG	41
TECHNICAL CONTRIBUTIONS	42
Restarting the TDS link multiplexer	42
IF transputers AND occam reliability := high	43
Semaphores at the transputer instruction level	46
More on the implementation of semaphores in occam	50
All about Trollius	54
A transputer simulator	66
PRODUCTS, SERVICES AND ANNOUNCEMENTS	71
Occam promotion initiative	71
Transputer education kit	72
The Queen's award for technological achievement 1990	74
TRANS-RTXc – the first true real-time kernel for transputers	76
Occam 2 and transputer engineering course	78
Digithurst multimedia system uses artificial intelligence	79
The static column TRAM	80
Helios now running standalone	82
Transtech new products	83
Quintek FAST9 allows portability of naval research system	85
Computer systems – science and engineering	87
REFERENCE	88
Contacts for related groups	88
Electronic grapevines	90
A word about names and numbers	90
Enrolment form	92
Special interest group chairmen	93
NATUG steering committee	94
Informal occam user group committee	95

Facilities to prepare this newsletter were provided by Oxford University Computing Laboratory; printed and distributed on behalf of the Occam User Group courtesy of INMOS Limited.