Innovative Technology for Computer Professionals

Computer

July 2005

Multiprocessor SoCs

IEEE

IEEE COMPUTER SOCIETY

http://www.computer.org

Innovative Technology for Computer Professionals
Computer
July 2005, Volume 38, Number 7

Cover design and artwork by Dirk Hagner

ABOUT THIS ISSUE

The emergence of application-specific integrated circuit and system-on-chip (SoC) manufacturing technologies in the 1990s laid the groundwork for a new era of post-RISC, configurable processors. Using the advanced development tools that are currently available, combined with the requisite software-development tools for that architecture, developers can now tailor a microprocessor core for specific application tasks in minutes, a shockingly brief time relative to the time spent designing processors and their associated development tools in prior eras. In this issue, we look at a reconfigurable MPSoC emulation platform from STMicroelectronics, ARM's instruction set architecture, Tensilica's configurable processors, and EEMBC's DENBench suite of digital media benchmarks.

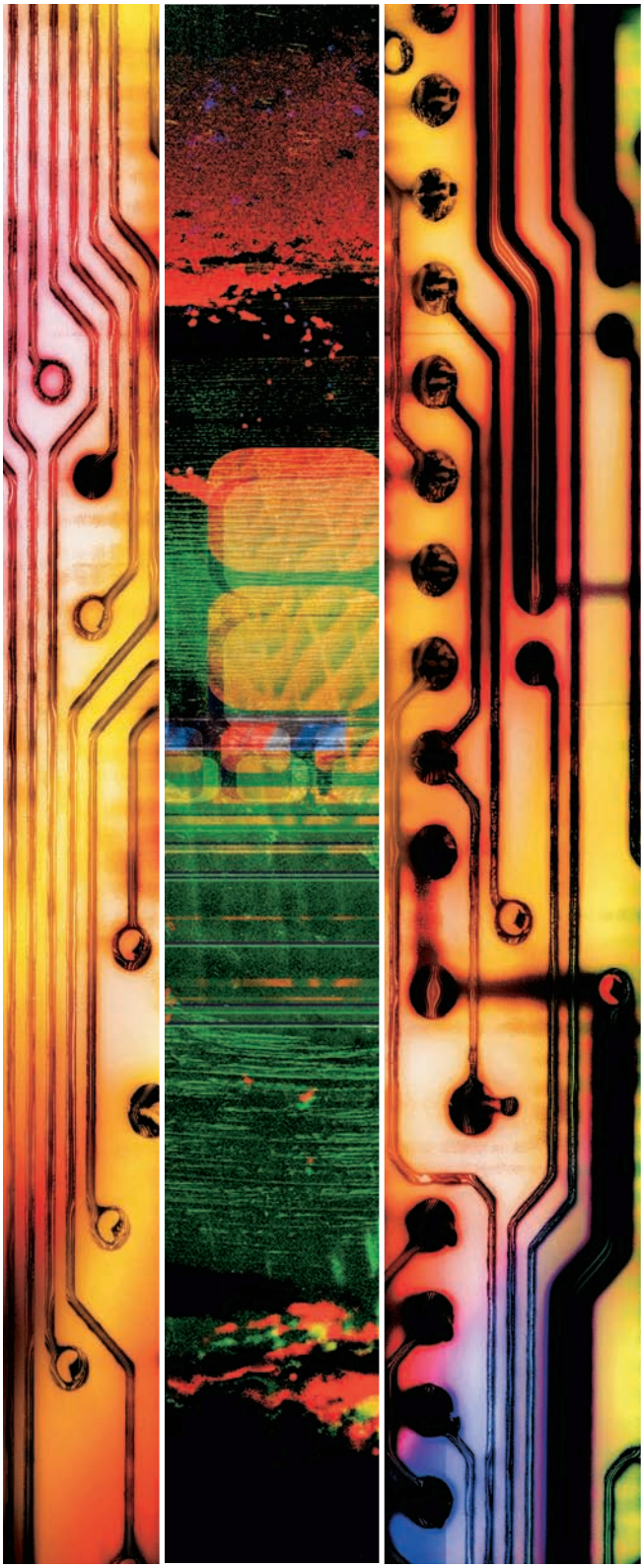
COMPUTING PRACTICES

COVER FEATURES

GUEST EDITORS' INTRODUCTION

Computer

IEEE Computer Society: http://www.computer.org
Computer: http://www.computer.org/computer
computer@computer.org
IEEE Computer Society Publications Office: +1 714 821 8380



NEXT MONTH:
Ultimate Display Technology

Membership Magazine of the

IEEE

IEEE COMPUTER SOCIETY

Innovative Technology for Computer Professionals
Computer

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society. **IEEE Headquarters**, Three Park Avenue, 17th Floor, New York, NY 10016-5997; **IEEE Computer Society Publications Office**, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; voice +1 714 821 8380; fax +1 714 821 4010; **IEEE Computer Society Headquarters**, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903. IEEE Computer Society membership includes $19 for a subscription to *Computer* magazine. Nonmember subscription rate available upon request. Single-copy prices: members $20.00; nonmembers $94.00.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Canadian GST #125634188. Canada Post Corporation (Canadian distribution) publications mail agreement number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8 Canada. Printed in USA.

ARTICLE SUMMARIES

Securing Wi-Fi Networks
pp. 28-34

Kjell J. Hole, Erlend Dyrnes, and Per Thorsheim

As Wi-Fi networks have become increasingly popular, many corporations have added Wi-Fi access to give employees easier access to corporate data and services. Although IT personnel control Wi-Fi access points in the corporate network, they cannot control access points in home networks. These networks thus give hackers new opportunities to gain unauthorized access to corporate computer systems and their data.

The results of an investigation conducted to assess the security level in Wi-Fi networks in Bergen, Norway, provide a context for analyzing some popular wireless security techniques and for offering suggestions on how to better protect these networks from hacking.

Parallelism and the ARM Instruction Set Architecture
pp. 42-50

John Goodacre and Andrew N. Sloss

The ARM reduced-instruction-set computing processor has evolved to offer a family of chips that range up to a full-blown multiprocessor. Embedded applications' demand for increasing levels of performance and the added efficiency of key new technologies have driven the ARM architecture's evolution.

The ARM team has used the full range of computer architecture techniques for exploiting parallelism, including variable execution time, subword parallelism, DSP-like operations, thread-level parallelism and exception handling, and multiprocessing.

The ARM architecture's developmental history shows how processors have used different types of parallelism over time. With its foundation in low-power design, the new ARM11 MPCore multiprocessor can bring low power to high-performance designs, which show the potential to truly change how people access technology.

Configurable Processors: A New Era in Chip Design
pp. 51-59

Steve Leibson and James Kim

Designers can use advanced development tools to tailor a microprocessor core for specific application tasks and generate the processor's register-transfer-level description. They can also generate all the requisite software-development tools for that architecture in minutes. This entire process takes a shockingly brief time relative to the time spent designing processors and their associated development tools in prior eras.

With automated tools, designers can focus on system architectural issues to achieve performance goals rather than spending time designing individual functional blocks within the SoC.

The configurable processor represents the next evolutionary step in microprocessor development, paving the way for new and interesting architectures that employ multiple, heterogeneous processor cores and exploit the qualities of advanced semiconductor lithography.

An Open Platform for Developing Multiprocessor SoCs
pp. 60-67

Mario Diaz Nava, Patrick Blouet, Philippe Teninge, Marcello Coppola, Tarek Ben-Ismail, Samuel Picchiottino, and Robin Wilson

The opportunities that nanometer technologies provide, combined with the consolidation of platform-based design approaches, have driven the evolution toward multiprocessor architectures, and the network-on-chip paradigm suggests new methods for designing and verifying embedded systems.

Clearly, a pure software simulation platform can't provide the performance required for developing multiprocessor system-on-chip designs. What's more, a main design risk for today's systems is the architecture, which developers must validate early in the design cycle because it has the biggest impact on system dimensioning and performances.

The authors describe an approach that introduces concurrent hardware and software engineering early in the development process and uses low-cost emulation facilities. Their approach extends the emulation used for verifying application-specific integrated circuits and application-specific standard product devices to multiprocessor architectures. They plan to introduce this platform in consumer and telecommunications product development to increase software and hardware engineers' productivity, which will reduce development time and costs while ensuring design and product quality.

Evaluating Digital Entertainment System Performance
pp. 68-72

Markus Levy

Digital entertainment systems have become the driving force behind the expansion of the semiconductor market, outstripping even PCs. In 2003, for example, smart phones represented about 3 percent of the 500 million mobile phones sold worldwide, with analysts expecting their sales to grow at triple-digit year-over-year rates.

More than half of the 600 million mobile phones sold in 2004 included a color display and digital camera. The implementation of more advanced features such as accelerated 2D and 3D graphics, videoconferencing, mobile multimedia, and games has raised performance requirements. The same holds true for other digital entertainment devices.

Rapid advances in semiconductor technology, microarchitectures, and embedded systems have made the adoption of these features possible. As a result, software complexity will continue to increase to keep pace with overall system complexities.

Performance and quality provide good starting points for evaluating a digital entertainment system, but energy consumption is an equally important metric.

ICSM 2005

Attend the 21st International Conference on Software Maintenance (ICSM), the world's foremost conference for software and systems maintenance, evolution, and management. This conference looks back upon a 22 year history of cooperation between industry and research in solving the problems of sustaining and enhancing existing software systems. It combines research results with reports from the field and tool demonstrations. Topics range from source analysis and reverse engineering to maintenance process modeling and maintenance cost estimation.

Continuous Evolution is the theme. It emphasizes that complex software systems are never finished. Once they go into production, they enter a phase of continuous evolution, which carries on as long as there are new requirements to be fulfilled. The task of maintaining a system in production, at the same time renovating and enhancing it, has been compared to repairing and reengineering a plane in flight, at the same time lengthening the wings.

The ICSM conference will give you the chance to meet with professional colleagues from all over the world and to share your experience with them in a congenial atmosphere. You will be able to attend industrial sessions with reports on real life projects, to study at tutorials by experts on maintenance topics, to hear keynote speeches from renowned experts, to see tool demonstrations, to take part in panel sessions, and to listen in on the latest research reports.

Don't miss this great opportunity to enhance your knowledge and enrich your skills, and to learn more about a country with a fascinating culture and an interesting past. A csardas evening in the wine cellars of the Buda hills and a trip to the puszta will be offered to conference attendees. The conference itself is located on an island in the Danube river between the twin cities of Buda and Pest.

Call for Participation

International Conference on Software Maintenance

International Conference on Software Maintenance 2005

Budapest, Hungary,
26-29 September, 2005
www.inf.u-szeged.hu/icsm2005

REGISTER

ONLINE AT:
http://www.inf.u-szeged.hu/icsm2005/

BY EMAIL TO:
Cserep.andi@mail.gatesgroup.hu

BY FAX TO:
#36-1-214-9316

IMPORTANT DEADLINES:

Early Registration closes:	August 30, 2005
Hotel rooms to be booked by:	August 25, 2005

CONFERENCE:

Tutorials and Industrial Session:	September 26, 2005
Regular Conference:	September 27-29, 2005

TUTORIALS:

"Developing Supportable Enterprise Information Systems"
by Leszek Maciaszek
"Using Metrics to improve Maintenance Testing" by Alfred Sorkowitz
"Object-Oriented Reengineering - Patterns & Techniques"
by Serge Demeyer, Stephane Ducasse & Oscar Nierstrasz
"60 years of Software Maintenance"
by Nicholas Zvegintzov & Girish Parikh

KEYNOTES:

Ian Sommerville on "Construction by configuration"
Girish Parikh on "Software Support, Management and Evolution"

GENERAL CHAIR:

Harry M. Sneed, ANECON GmbH, Vienna, Austria,
Harry.Sneed@t-online.de

PROGRAM CO-CHAIRS:

Tibor Gyimóthy, University of Szeged, Hungary,
gyimi@inf.u-szeged.hu
Vaclav Rajlich, Wayne State University, USA
rajlich@cs.wayne.edu

The patron of the conference is E. Sylvester Vizi,
President of Hungarian Acadamy of Sciences.

REF

IEEE COMPUTER SOCIETY

A PACKING LIST

Before Bob Colwell sets out on his missionary work to explain engineering to the humanities community (At Random, "Frames of Reference," June 2005, pp. 9-11), he should include the following in his packing list:

- a book on reasoning,
- a history of the 20th century, and
- a subscription to the *Journal of the History of Ideas*.

The New Age mystic and the school board official that Colwell cites as representatives of the humanities are as abhorrent to those of us in the "difficult" sciences as they are to our colleagues in the "hard" sciences.

Colwell appeals to the common prejudice of technologists that the humanities have no standards, hence, anything goes. That appeal is furthered by an inadequate sample that would not pass muster in any technology discussion.

Is scientific illiteracy a problem? Undoubtedly, but then so is humanities illiteracy. What else would explain representatives of the hard sciences writing claptrap like the Bible Code books? It is not good science, much less good humanities work.

But it would be unfair, not to mention inaccurate, to characterize all technologists on the basis of a few carefully chosen examples.

Colwell's claim in his conclusion that people in the technology business must realize they are "in the service of truth, an absolute truth that must be guarded" is contrary to the history of the 20th century. I will skip the usual list of racial and social outrages perpetrated in the name of science and service of "absolute" truth. No doubt Colwell would reply that those were betrayals of the scientific process, and I would readily agree.

The problem is that when any group—technologists, humanists, or any groups ignorant of one or both—decides that it has a lock on absolute

or even provisional truth, bad consequences follow.

The information overload issues currently reported with such breathless concern in the major news outlets and more than a few scientific circles have been encountered before. For example, the January 2003 issue of the *Journal of the History of Ideas* was entirely devoted to the topic of "Early Modern Information Overload." Realize that "early modern" in this context refers to the time period 1550-1750.

While the solutions developed at the time did not rely upon the transitor and its now distant cousins, the underlying principles were not all that different from modern information systems. The point being that if we avoid the obviously outlandish on either side, we can learn a great deal by studying both the hard and the difficult sciences.

Colwell starts from the position that "I'm right and you have been too lazy to learn why, so sit down and listen," which will not excite a lot of interest in his target audience. A better strategy would be to try to understand the other frame of reference before deciding that it is all that different from your own.

Will this strategy mean reading unfamiliar material and having to master new terminology? Yes, but it is also potentially rewarding in terms of new insights into your own discipline.

If the arts/science gap is going to be bridged, it will be by participating in both communities, not by missionaries bringing enlightenment to the savages.

Patrick Durusau
Covington, Ga.
patrick@durusau.net

Bob Colwell responds:

In my June column on "frames of reference," an idea that it is very difficult to communicate across different frames, I juxtaposed science and religion and science and the arts as pairs of representative differing frames. To find further evidence for the difficulties of communicating across frames, it isn't necessary to look any further than Patrick Durusau's letter.

Within the scientific frame of reference, most of its practitioners (including me) presume that an absolute truth exists. Scientists spend their lives looking for it; engineers spend their lives trying to stay close enough to it that their designs will work as intended. Durusau's computer works because these engineers succeeded.

The electrical charge on an electron is the same no matter who measures it, no matter where, and no matter what attitude or beliefs the measurer may have. Most of Durusau's letter seems aimed at the idea that science and the humanities are duals. In many ways they are, and thinking about those similarities may be enlightening to both camps.

But there is a basic way in which they differ, and Durusau has inadvertently highlighted it: Those in the sciences have a different standard for what they consider truth. When people in the humanities say that science has no monopoly on truth, I agree. But in an important sense, we're talking past each other. We don't mean the same thing; we're just using the same word.

Both sides ought to consider that each is using it within its own frame. That was precisely why I bridled at the mystic in the movie—she was using what she meant by "truth" as if it meant the same thing as when a scientist uses it, thus propagating an insidious logical error.

Durusau missed one other major point of the article: It was aimed precisely at *Computer*'s readership, and not at anyone from any other frame of reference. I therefore intentionally used the terms and context that are appropriate to that frame. Were I to have written a similar article for a humanities journal, substantial translation would have been required so as to reach that readership

without giving offense. That was the point of the article.

CAMP A INSIGHT

Regarding Bob Colwell's June column, I think Camp A is actually all right with Camp S. We need a different label than "art" for the other side.

I say this from my experience teaching a new course at the Maryland Institute College of Art titled "Foundations of a Scientific World View." It's a very selective school—these are top-notch kids with great artistic gifts and motivations, and they're no slouches when it comes to traditional academics either.

I also teach a course for graduate engineering students at Johns Hopkins titled "Logical Foundations of Computer Science." The art students have an unfair advantage, being full-time undergraduates versus the part-time graduate engineering students, but I'd have to say the art students are more inquisitive and open to a range of ideas compared to the engineering students. And the art students are very rigorous, too. The art faculty is also very inquisitive, open, and rigorous.

Maybe the opposite camp from Camp S is Camp B, for believers. Camp A is chock full of questioners. They seem to love a radical challenge to their preconceived notions.

Richard (DJ) L. Waddell Jr.
Laurel, Md.
richard.waddell@jhuapl.edu

Bob Colwell responds:

I can readily testify that most artists I've known have exhibited as much discipline and rigor as most engineers. But you know how stereotypes go: Artists are creative; engineers are drudges. I don't buy that one, either. Engineers are among the smartest people I've ever met (including all my science friends who are also very bright), and smart people are easily bored. I think people operating at the limits of human intellect and feasibility will always be creative, artist or engineer, and probably to the same extent.

In my June column, I was primarily trying to find different frames of reference that the readers—who are mostly in the scientific camp—would readily understand, without inadvertently extending any stereotypes that should not be propagated. That's why I didn't address creativity, intelligence, inquisitiveness, and so on. I'm not trying to compare and contrast, per se; I'm only trying to show that the idea that differing frames cause differing interpretations of ideas and words is a useful meme.

My motivation is that within the next two to three decades, the probability is quite high that the world will face some fundamental challenges, of types that have never been seen before. Global warming, caused by the wholesale dumping of carbon into the atmosphere, could cause massive, unpredictable changes to the climate worldwide.

To what extent should we make sacrifices today on behalf of future generations, and based on what data and climate models? When the stakes are this high, we had better get this right.

Avoiding such catastrophes will require a reasonably deep understanding of many technical issues. That people not already embedded in the scientific realms must be bright, enthusiastic, and open to ideas is necessary but not sufficient.

The science folks who have studied this issue must find ways to get the information across to their Not Camp-S compatriots, or making communal intelligent decisions will be impossible. That is what I meant when I said the Camp-S folks must bridge the gap.

ARITHMETIC OPTIONS

While my interest in reading "An Open Question to Developers of Numerical Software" by William Kahan and Dan Zuras (Standards, May 2005, pp. 91-94) kept up till the end, my perturbation grew from nothing to very great.

The authors state that the IEEE 754 Revision Committee's work is directed "into areas of expression evaluation, compiler optimizations, and other language issues that were considered outside the scope of the 1985 standard." Although the article opens with a general description, it is evident that the authors pose the open question because the committee members "wish to get rid of traps and of sNaNs if [they] can." The concluding remarks confirm that this is the primary concern, as does the e-mail address given for sending comments: snans@nonabelian.com.

The authors mention two excellent objectives: "Revisions of substance … that would most help make modern floating-point programming easier and more reliable, efficient, and portable" and not to "make any change so sweeping it invalidates the operation of most of the world's computers."

What the authors seem to overlook is compatible extension. Defining standards for arithmetic on double (not double-length) numbers as well would best meet their worthy objectives.

There could be four modes of use: extended precision, combined operations on number pairs, operations on intervals, and operations on complex numbers—the last two being of most benefit in computation.

A method of interrogating whether the double arithmetics are available would make transition to the extended standard much easier.

Although the more popular programming languages do not as yet have types available for such extensions, the arithmetic could come into prompt and effective use in standard library code.

The availability of full support for interval arithmetic would benefit technical computation and promote wider use of the proven validated numerics techniques. The availability of full support for complex arithmetic would greatly improve both technical com-

putation and graphical computation, which is of such commercial significance and magnitude that chipmakers might well be persuaded to adopt a new extended standard much earlier than if it merely benefited engineers.

I am not a numerical software developer, and many such developers would be better able to judge the benefits of this kind of extension. If I am wrong, so be it. But if I am right, and if the benefits are as great as I imagine, this is an opportunity for experts to offer the Revision Committee their opinions and their help. If the committee is to extend the standard in this way, they will need the best help they can get.

Neville Holmes
University of Tasmania
Neville.Holmes@utas.edu.au

The authors respond:

Neville Holmes is quite right in stating that the primary goal of our article was to get feedback on the narrow issue of signaling NaNs.

The IEEE 754 Revision Committee has considered many of the extensions Neville mentions, along with others. However, doubled-double generally is an unreliable arithmetic to use for computation.

Suppose a C++ program using float and double variables and constants has been proved to work well, but greater accuracy is desired. One way to achieve that is to promote all double variables and constants to doubled-double (most likely called "long double"), promote all float variables and constants to double, appropriately shrink tolerances controlling iteration convergence, add terms to truncated series, and so on.

Would the promoted program work as well as the old, but twice as accurately? Not necessarily. The promoted program could suffer from mysterious rare malfunctions none of which could afflict the code if quadruple had been used instead of doubled-double.

This is why the committee has chosen to standardize the honest 128-bit quad format already in common use. At least one chip designer has announced full hardware support for quadruple at full speed.

The IEEE 754r committee has avoided attempting to specify complex arithmetic for three reasons.

First, C99 does a good job. It specifies an imaginary type and then defines the complex type as a formal (unevaluated) sum of a real and an imaginary. Doing so avoids several nasty anomalies in the complex type as Fortran and some packages used in C and C++ specify.

Second, as good as the C99 specification may be, it still suffers from some conundrums. For instance, multiplication that handles complex infinities well is too slow for matrix multiplication.

Correctly rounded complex multiply is probably too much to ask even from systems with a fused multiply-add but not quadruple in hardware. Correctly rounded complex divide still costs too much.

Third, nothing IEEE 754r can say will change entrenched complex arithmetic practices, but attempts to change them might well jeopardize acceptance of the rest of IEEE 754r. Maybe some day.

We have looked into interval arithmetic and a standard set of transcendental functions. Our reluctant conclusion is that the state of the art is not yet up to standardizing these functions in a way that our posterity would surely find satisfactory.

Some things need to be standardized now:

- control of expression evaluation,
- alternate methods of exception handling, and
- aids to debugging.

Our work is not yet finished. The committee's deliberations are open to anyone who wants to participate (grouper.ieee.org/groups/754/), including those who are not members of the IEEE.

For details about these and other issues see www.cs.berkeley.edu/~wkahan.

William Kahan and Dan Zuras

COMPUTATION AND COMMUNICATION

In "Socially Aware Computation and Communication" (Mar. 2005, pp. 33-40), Alex Pentland discusses some interesting cutting-edge ideas for building machines to understand human behaviors in terms of "quantifying social context in human communication."

In addition to being the underpinnings of mental partnerships, nonlinguistic signals such as body language, facial expression, and tone of voice are what make humans the most complicated animal. However, how well an expert system could understand and interpret such signals would depend on its capacity to make inferences or draw conclusions. It is not a question of how to build such an expert system, but rather whether the system will work.

In general, an expert system consists of two important components: a knowledge base and reasoning. This base consists of factual knowledge and heuristic knowledge.

Factual knowledge is a vast collection of widely shared and commonly used information. How people use this collection of knowledge depends on multiple factors such as their age, education, culture, gender, and so on. In addition, the same behavior could have a different meaning in a different situation or a different culture.

In some cases, knowledge is incomplete or uncertain. While it's possible to apply the concept of "fuzzy logic," assigning an associated confidence factor or weight, these scenarios might excite a human's imagination but probably not a machine's.

Heuristic knowledge is useful in testing the efficacy of an intelligent system because it requires making judgments, but this could cause plausible reason-

ing to be ineffective. For example, if people do a good job of pretending, how would forming a line of reasoning using either forward chaining or backward chaining of IF-THEN rules handle interpreting their behavior?

That is what makes simulating the human brain extremely difficult, and I believe no machine can ever replace it.

As a side note, if researchers could build a machine for making important decisions like finding a mate, getting a job, negotiating a salary, or finding a place in a social network, would it become a hindrance to human thinking?

Although we don't know when such a machine might become available, and it seems unlikely that it would be effective in dealing with the real-life situations, I fully support the spirit of this wonderful pursuit of the concept of building intelligent systems.

Hong-Lok Li
Vancouver, BC
lihl@ams.ubc.ca

EMBEDDED ENTERTAINMENT

Readers of my article on smart projectors published in *Computer*'s January 2005 issue ("Embedded Entertainment with Smart Projectors," pp. 48-55), might be interested in finding information about a related project. The authors of "Making One Object Look Like Another: Controlling Appearance Using a Projector-Camera System" (M.D. Grossberg et al., *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, IEEE CS Press, 2004, pp. 452-459) describe a projector-camera system that addresses the problem of radiometric compensation on 3D surfaces with a different technical approach and application orientation.

Oliver Bimber
Bauhaus-University Weimar
bimber@uni-weimar.de

We welcome your letters. Send them to computer@computer.org.

JULY 1973

ELECTROCARDIOGRAPHY (p. 21). "The central issue is, of course, how we can provide more expeditious and efficient [electrocardiography] service. A computer can help us achieve this.

"We can quote from a publication entitled 'Computer Assisted Medical Practice: The AMA's Role' published by the AMA in 1971. 'It is reasonable to expect that government and industry will make broader commitments to develop further the applicability of the computer to medical practice. The ultimate responsibility for the effective utilization of computers, however, will lie within the medical community.' Correctly it is up to physicians to assure that efforts are successful. To do this they must begin to invite engineers into the clinical area. A case against computer electrocardiography can be made. One wonders who after even the shortest study would really want to make it."

INTENSIVE CARE (p. 29). "The application of technology and system engineering techniques to the delivery of health services has made possible the successful implementation of a computer-based system in the clinical care of patients during the crucial early hours following heart surgery."

"Routine, repetitive tasks which are well defined have been relegated to the system, enabling the nurses to devote more of their time to direct patient care. The computer automatically acquires the clinical measurements simultaneously from four patients each two minutes, displays and stores the current values, retrieves past data for review at bedside on command and periodically tabulates the data in hard copy form to be included in the patients' hospital records, relieving the nurses of nearly all measurement and charting chores."

SHEET PRINTER (p. 35). "Xerox Corporation has announced a new non-impact computer printing system that produces copy on 8½ × 11-inch, ordinary, unsensitized paper faster than a page a second, or up to 4,000 lines per minute."

"It is about twice as fast as standard impact-type computer printers now on the market … and does away with a need for the large, unwieldy paper and pre-printed forms on which computer print-out has been produced in the past.

"A key advantage of the Xerox 1200 computer printing system is the elimination of the bursting and decollating operations—removing carbon paper and separating continuous-form sheet—associated with impact printers."

DISPLAY TERMINAL (p. 36). "A new CRT Data Display Terminal announced by Lear Siegler, Inc. has been described as a 'breakthrough' in the area of cost vs. performance. Exhibiting an impressive list of capabilities, the new terminal has been priced at under the $1000 level in quantities and is listed at $1500 in single units, roughly half the cost of conventional devices."

"Performance-wise, the ADM-1 has capabilities matching and exceeding most conventional CRT terminals. The display format of the terminal is 960 characters (12 lines of 80 characters), using 64 alphanumeric US ASCII characters in a 5 × 7 dot matrix. An optional screen is also available with 1920 characters consisting of 24 lines with 80 characters each."

COMPUTER ANIMATION (p. 40). "The computer has moved past science and business and into the creative arts with the announcement of 'SynthaVision', a process that makes multi-dimensional color films, completely by computer, without requiring the existence of an original 'except in the mind'—with realism comparable to photographs of an existing object.

"Representatives of the nation's leading advertisers and their agencies, television and film production companies, scientists, educators and urban planners attended the first in a two-day series of presentations … They saw a 22-minute demonstration film and heard Dr. Phillip S. Mittelman, who conceived the process, say that with SynthaVision, 'you can now produce on film a simulation of almost any form or object imaginable. It can grow, shrink, change shape and size—anything you wish it to do—and no original is required, only your idea of what it should look like and what it should do.'"

LIGHTING CONTROL (p. 41). "At the birthplace of William Shakespeare, one of the largest computerized lighting control systems in modern theater has been installed in the Royal Shakespeare Theatre, Stratford-Upon-Avon, England."

"Each lighting state set up during rehearsal is recorded by the system. The data recorded are the voltage levels required to drive each dimmer circuit. The values are stored in the computer's memory for use during the actual performances.

"The computer is connected to a control console, output devices, and a casette recorder. The system scans the lighting controls, interprets the commands given to it, computes cross-fade data for each control channel—updating when necessary—and initiates control commands to the theater lights."

INTERNATIONAL BANKING (p. 41). "S.W.I.F.T. (Society for Worldwide Interbank Financial Telecommunication) has been legally incorporated as a non-profit making society and the board of directors held its first meeting in Brussels.

"The S.W.I.F.T. network will carry financial transactions for a consortium of international banks. Out of a total of 256 banks which have participated in the design phases of the project 239 have now become members of S.W.I.F.T. They represent the major commercial and central banks in 13 countries in western Europe as well as in Canada and the United States."

"The S.W.I.F.T. network is planned to come into operation in 1976. It will be a store and forward switching net-

work designed so that users with terminals of different speed and type can communicate with each other."

JULY 1989

SDI LETTER (p. 5). "In short, SDI can only be called reliable by tampering with the very notion of reliability itself. Not all of the conferences, speeches, statistics, or appropriations in the world will change this fact. SDI has no intellectual credibility—it is the 'creation science' of the engineering world. Blandly reporting the double talk from the latest SDI conference is no service to the public. Who, except those bellying up to the federal feed trough, can think that SDI is worth $30 billion in software development costs? As an engineer, I say, let's build projects that actually work. As a taxpayer, I say, SDI is an outrageous grab for the federal purse, bleeding money away from legitimate projects, both military and civilian. Let's stop the SDI boondoggle."

DYNAMIC SCHEDULING (p. 21). "Many features of the pioneering CDC 6600 have found their way into modern pipelined processors. One noteworthy exception is the reordering of instructions at runtime, or *dynamic instruction scheduling*. … Another innovative computer of considerable historical interest, the IBM 360/91, used dynamic scheduling methods even more extensively than the CDC 6600.

"As the RISC philosophy becomes accepted by the design community, the benefits of dynamic instruction scheduling are apparently being overlooked. Dynamic instruction scheduling can provide performance improvements simply not possible with static scheduling alone."

DESIGN RECOVERY (p. 36). "Software maintenance and harvesting reusable components from software both require that an analyst reconstruct the software's design. Unfortunately, source code does not contain much of the original design information, which must be reconstructed from only the barest of clues. Thus, additional information sources, both human and automated, are required. Further, because the scale of the software is often large (hundreds of thousands of lines of code or more), the analyst also needs some automated support for the understanding process.

"Design recovery recreates design abstractions from a combination of code, existing design documentation (if available), personal experience, and general knowledge about problem and application domains."

ASSOCIATIVE MEMORY (p. 51). "Researchers have understood the basic principles of storing and retrieving data by content rather than by address for about 30 years. Despite this relatively long incubation period, information has spread slowly from the academic arena, and the technology has not been available to produce a successful commercial product. As a result, many designers have not developed the skills to work with associative and content-addressable memories. However, VLSI technology has improved the feasibility of associative systems and overcome many implementation obstacles."

MULTIPROCESSOR SYNCHRONIZATION (p. 66). "The growth of multiprocessors is evidence of an increasing focus on achieving high program speeds through parallelism. One of the primary problems confronting designers of multiprocessors is to provide efficient synchronization methods. The concurrent execution of programs may be limited by the parallelism exhibited in the control mechanism and by the associated overhead. A family of effective synchronization concepts can aid in the design and construction of parallel programs. Although synchronization is a long-standing area of research, existing solutions must be readdressed in the context of specific constraints posed by general-purpose, multiple-instruction, multiple-data (MIMD) architectures."

100,000th MEMBER (p. 82). "Mark Funkenhauser, a 30-year-old Canadian researcher and graduate student, has been honored as the 100,000th member of the IEEE Computer Society.

"Funkenhauser became the society's 100,000th member on December 5, 1988. To commemorate that milestone, the society presented him a plaque May 17 during its 11th International Conference on Software Engineering in Pittsburgh."

OFFICE AUTOMATION (p. 102). "IBM says that its OfficeVision family of office-automation software is its first major System Application Architecture software application. The new software family reportedly provides integrated office functions across the OS/2, MVS, VM, and OS/400 operating systems."

APPLICATION DEVELOPMENT (p. 104). "Oracle has added the CASE Generator to its computer-aided systems engineering family of application development tools. According to the company, CASE Generator automatically generates portable applications directly from design specifications.

"CASE Generator receives definitions about an application's database tables and program module definitions from CASE Dictionary and translates the information into functional applications using SQL Forms, the company's fourth-generation development tool. The resulting applications reportedly enforce all constraints and validation criteria in CASE Dictionary. They support lists of valid values, help and hint text, and automatic synchronization of data from multiple database tables."

Editor: Neville Holmes; neville.holmes@utas.edu.au.

Judging Science Fairs

Bob Colwell

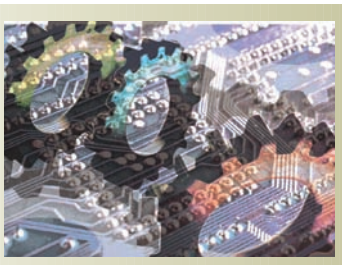
Arbitrary subjectiveness drives me crazy. I'd never have made it as a figure skater, for example. Apart from the requisite skill at skating, which disqualifies me right away, you would have to restrain yourself from physically attacking the judges after a competition, no matter how illogical, biased, self-serving, or obtuse their collective decision seems to be. And some randomness is built into the situation: "Artistic merit" can never be objective.

Auditioning for a play or an orchestra is much the same. Professionals have told me that even when a trumpet player is auditioning "blind"—behind a screen—after only a few notes, the judges will already know who he is. Does this bias their decisions? Well, they're human; if you're auditioning, it would be wise to expect subjectivity. Hope that you never offended them.

ENCOUNTERING CRYBABIES

I once asked an NBA referee what it was like to call a game with flamboyant figures such as Dennis Rodman on the court. Surely it would be easier and yield a fairer overall result if the ref didn't have to consider the actual personalities involved?

The referee agreed, and without referring to any particular player, said that some of them were much more insistent than others about referees calling fouls. "Crybaby" was the word he used, and he said that the truly great players were never in that category. He cited Michael Jordan as his favorite—someone who never demanded special treatment, even when he knew a call had been blown. (As an aside, the ref said, "Hey, the players make mistakes, and so do we.")



Engineering must work or it's of little value, even in a science fair.

This ref said that the crybabies don't accomplish what they hope for because by calling attention to themselves, they actually increase their chances of getting caught. Since they weren't exactly ingratiating themselves with the refs, any calls that could legitimately go either way were in greater danger of going against them.

INCENDIARY TOPICS

We're told to avoid discussing religion or politics with people we don't know well because the chance of inadvertently generating an emotional tirade is so high. There's another conversational topic with the same incendiary potential: tastes in music.

Apart from disco, which most people except John Travolta, Donna Summer, and two of the three Bee Gees hated, it's extremely difficult to have a mutually beneficial conversation about, say, rap music's contribution to human society. It's amazing to me that people care so much about this and to observe how subjective their judgments about music really are.

It's not just culture—education matters too. You can learn to appreciate some kinds of music by taking into account the composer's intent, the environment, and the overall context. But in the end, each of us reserves the right to pass judgment on whether a particular aural offering is good or not.

THE BEST

A perennial favorite topic in a classical guitar e-mail list is the question of which guitar is The Best.

If you're new to this topic, you might think that question has at least some hope of being answered rationally. For example, we might try to sneak up on it in several ways. What kinds of guitar do the finest players in the world use (never mind the battle over who those players are)? Maybe they all use the same kind, and we could narrow our search to just those. Which guitars cost the most? Surely there should be a correlation between price and quality.

If you polled all the best luthiers (there's that "best" thing again), would there be a pattern to their answers? Have any guitar competitions been held to try to shed light on this question? What if we actually measured a sample set of high-end guitars with appropriate audio equipment?

There are hundreds, possibly thousands, of classical guitar makers in the world. There are arguably fewer than 100 in the elite category of players that includes John Williams, Julian Bream, David Russell, and Elliott Fisk. And yes, there is a pattern to the guitars they play—it's clearly not a random linear distribution.

Published by the IEEE Computer Society

For example, there appears to be a preference for the new carbon-reinforced "tops," which are reputed to generate more volume, a good thing for filling concert halls when playing an instrument with an intrinsically small voice. But are those new instruments sacrificing all-important tone to achieve that volume? If they are, is that a necessary price to pay, or will luthiers learn to get both? Nobody knows. And what we do "know" is … subjective.

SCIENCE'S OBJECTIVE UNIVERSE

If there's just one thing that the scientific paradigm got right, it was to automate the process for filtering ideas and theories before their general acceptance.

It doesn't matter what you think of me, or I of you. If your idea explains reality better than mine, then the scientific establishment should prefer yours.

In reality, of course, humans do the judging, and in any given case, they might get it wrong. The NBA ref must struggle to be fair when an overpaid crybaby is trying to do his job for him; the music audition judge must resolutely shut out any context that might interfere when she realizes who is playing on the other side of the curtain. And when you're refereeing a paper, you must constantly watch for your own biases, especially for submissions from rival schools or rival researchers.

What differentiates the scientific process from these other subjective judgments is our implicit retesting. We may get it wrong once in accepting or rejecting a paper, but over time, other people will retest the ideas in other contexts. Ideas that make it through that gauntlet will have been honed, shaped, and refined in the process, and in general, any personal biases will be filtered out.

There are corner cases to this nice, ultimately objective world. One of them is judging science fair competitions.

HIT ME WITH YOUR BEST SHOT

Real-world designs are analyzed and evaluated by rival engineers, industry analysts, buyers, and in the case of microprocessors, more Web sites than you would ever expect. Usually, these evaluations are done on a relative basis: Which browser is better, Internet Explorer or Firefox? Which operating system is better, Windows or Linux?

The evaluators take measurements, render graphs, and draw conclusions. These reviews are brutal, and they often make the product designer feel as though his children are being gratuitously and maliciously attacked.

Implicit retesting differentiates the scientific process from other subjective judgments.

The more intrepid evaluations will also try to measure a real-world product in absolute terms. Your new laptop may exhibit an 8-hour battery life, and the best of the competition only five hours, but if some particular application requires three days, then your product will still be judged as lacking—as indeed it should be.

As a designer, I may not always enjoy these analyses, especially if I think they're unfair or biased. But I agree that real-world products should be held to the highest attainable standards.

JUDGING STRATEGIES

While the best projects at a world-class science fair like ISEF can hold their own with published research from anywhere, it isn't reasonable to expect the best engineering projects to do the same. There are many reasons for this, but one of the most important is that science has a lower barrier to entry: Individuals working alone can still have a good idea in math or science and have enough wherewithal to convince others it has merit.

Engineering, in contrast, also starts with ideas, but it usually also requires building something. Brilliant students are as creative as experienced engineers in industry, but they lack the experience to know how best to apply it. They also seldom have access to an industrial design team's resources for realizing their idea, and they can't hope to compete with a competent design team in terms of design-hours, tools, and experience. Moreover, engineers usually work on projects that are expected to yield a lucrative outcome—there's money involved and therefore there's strong motivation.

How then can we fairly judge a science fair effort in engineering at the high school or college level? If you compare it against standard industrial practice, it will inevitably come up short. But if you only compare the projects against the other submissions, what feedback should you give the submitters about their work's merits and drawbacks?

I've judged many such events. There are a few things that are said at every closed-door judging session: "These are only high school (or college) submissions; you can't expect them to hold up to professional standards." Or "What were *you* doing in high school?" Or "Aren't you being way too hard on these poor kids?"

All things considered, I believe the best strategy is to judge submissions both ways: 1) relative to each other, and 2) absolutely, against the highest standards the judges can conceive.

Why judge relatively?

The point of a high school science fair or college-level science competition isn't to generate new knowledge or fundamental contributions to the engineering experience base, although that does occasionally occur. The point is for the students to have fun with science and engineering, further their knowledge and ability, and remain engaged with the technical process.

Kindergarten through eighth-grade science fairs are noncompetitive. I have mixed feelings about that. Especially for the youngest exhibitors, it would make little sense to try to identify "the best" because it would be counterproductive

in terms of making kids like science (it's not much fun to lose). Also, to be frank, even the best of these tend to be not very good, so overly rewarding them would (within a few years) turn into a mixed message for the budding scientist.

The challenge is to give the students who are really trying enough useful feedback so that they learn and grow, without discouraging them about how far they have to go.

High school and college exhibitors want to know how they fared versus their peers. They want to know, "Am I any good at this science/engineering thing? Can I do it?"

I try to draw these students out on why they picked the topic, the ways they considered resolving it, and why they made their choice.

It's their reasoning ability, their ability to abstract out the essential elements of the puzzle and exclude whatever can be safely excluded, that I'm looking for. When I find that thread through their thinking, I highlight it and suggest that they've demonstrated that they possess the one trait no scientist or engineer can do without: the ability to make reasoned abstractions.

Searching for the best submissions helps to illuminate this process to all participants. There are degrees of thoroughness, shades of gray in how diligently students can strive to eliminate uncertainties from the work, a wide range of how much background research they can find to buttress their conclusions or help justify the study.

Only experience can inform the student's intuitions about how much is enough, and the science fair process is one such experience. Judging student projects against one another helps each student see where the judges believe those lines should be drawn.

Why judge absolutely?

At the dawn of the nuclear era, the military was enthralled with the idea of atomic submarines that could stay submerged for a very long time, something the battery- and diesel-powered subs of World War II couldn't do. They asked if the same trick could be applied to aircraft: Was it possible to keep a bomber in the air for several weeks at a time, powered by nuclear fission?

Fortunately, the answer turned out to be no (would you want nuclear reactors flying over your head?). The materials required to shield a nuclear reactor for nearby humans turn out to be very heavy. Not a problem for submarines and ships, but an insurmountable issue for aircraft.

Both relative and absolute judging are essential to achieving the right balance of fun and education in science fairs.

Imagine a science competition in which the best three engineering submissions turned out (by coincidence) to be different designs for atomic aircraft. After intense scrutiny of these projects, a consensus ranking by the judges might well emerge. But the judges might also conclude that none of the three submissions was, in fact, workable.

Rough edges on a prototype are understandable and to be expected—engineering submissions that could encounter economic or fabrication challenges could be wonderful despite obvious flaws in the prototype being displayed. Professional engineers everywhere will recognize that truth; they live it every day.

Economic issues can be overcome with high sales volume (witness LCD displays); better tools or specialized manufacturing techniques (blue LEDs, for example) can overcome fabrication challenges. But if the project's fundamental premise is flawed, it is doomed in a way that no pretty display poster can possibly fix.

Judges have a responsibility to make this clear to the project submitters. It's not enough to be the best of a bad lot. Engineering must work or it's of little value, even in a science fair.

I have seen several instances in which perpetual energy machines made it to ISEF. In each case, the combination of bright, engaged students and the clear evidence that they aren't getting appropriate guidance from their teachers (and the judges they somehow got by at lower levels of competition) was profoundly depressing to me.

Should every project that even appears to rely on—or produce—perpetual motion be automatically excluded from ISEF? No, I wouldn't go that far. As a matter of practicality, the US Patent Office is within its rights to auto-exclude such things from consideration for patent protection. But science as a whole must not.

However, what must be made clear to such students is the fundamental, inescapable truth that "Big Claims Need Big Proofs." If you're going to claim perpetual motion, or cold fusion for that matter, you need compelling evidence. Your work is going to undergo extraordinary scrutiny, and thorough preparation for such an ordeal is much better done beforehand and in private, rather than after you've staked the claim.

THE BURDEN OF PROOF

Turning science on its head is among the most revered accomplishments to which students can aspire—Albert Einstein is the patron saint of such efforts. But the burden of proof for such things is extremely high.

More commonly, in the best tradition of engineering, students will find something in their own environment that they believe they can improve. The family car, for example: It breaks down, it gets into accidents, its driver may be impaired or lost, and the interior sometimes gets hot enough to kill pets or children.

The students imagine ways of having the vehicle sense its surroundings, automatically communicate with other vehicles, and autonomously contact

the nearest hospital upon detecting that the airbags have deployed. They propose "black boxes" to collect information about automobile collisions. They work out bandwidths, power levels, cost, packaging, and programming. The best students even prototype the system and try it out in the lab. When it functions as expected, they declare victory and make their poster for the competition.

A real-world engineering team would consider things far beyond the obvious technical basics. The automotive environment is a very hostile place for electronics—everything must work correctly, every time, in Fairbanks, Alaska, in late January and also in Death Valley, California, in August. Those temperature extremes are much wider than normal commercial-grade electronics parts are guaranteed to handle.

Because they're mobile, cars can also drive close to antennas pumping out kilowatts of radio-frequency power. Keeping electronics working correctly in the face of this onslaught requires careful attention to grounding, shielding, and signaling.

More subtle, for most students, are the not-quite-science issues that surround a real-world design. How do you reliably detect that a driver is getting drowsy? Is there a single reliable way to detect sleepiness? Head position? Drooping eyelids? The car wandering across lane markers? Having disco come on the radio and not quickly changing the station?

What should the system do if it determines that the driver has become drowsy—find a heavy-metal station and turn up the radio? If the system malfunctions and begins to annoy the driver, is it possible to temporarily turn it off?

How would you know that a scheme for having an automobile communicate with a hospital works? Will it routinely ping nearby hospitals, or will it just do a power-up self-test when you start the engine? Is the Internet reliable enough to incorporate into such a scheme?

Perhaps most difficult for students are the legal issues, which can differ radically from one country to the next. How do you design fail-safe automotive products? You already know that a black box recorder for a car must be designed to survive a high-G crash and ensuing fire. But how will you keep it from being hacked to make it seem as though the other driver was at fault? Who owns the information in that black box, in a legal sense? How trustworthy would the data be considered in a court of law?

I wouldn't expect students to have convincing answers to these questions, but if they're going to design for the automotive arena, they must at least consider them.

Only by illuminating the gap between a given project and what the real world would require will students be able to recognize the value of their contribution and the size of the learning curve they have yet to climb.

The best students already operate at the highest levels of intellect; we should honor them by treating them as respected colleagues and never be patronizing or condescending about their efforts.

The optimum balance between relative and absolute judging will differ between the best students and the rest, but both kinds of judging are essential to achieving the right balance of fun and education. ■

***Bob Colwell** was Intel's chief IA32 architect through the Pentium II, III, and 4 microprocessors. He is now an independent consultant. Contact him at bob.colwell@comcast.net.*

Eclipse Becomes the Dominant Java IDE

David Geer

Since Sun Microsystems introduced Java in 1995, proponents have sought ways to boost the technology's fortunes. One approach has been to create an integrated development environment that would make working with Java easier.

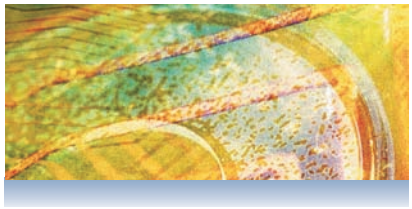
Supporters hoped an IDE would make Java more competitive with Microsoft's popular Visual Studio .NET, which provides an environment for integrated, easy-to-use software tools that appeal to the many business-application developers who aren't hard-core programmers.

This has set off a battle among several Java IDEs, including Borland's JBuilder, Microsoft's Visual J#, Oracle's JDeveloper, and Sun's NetBeans.

One contender has been Eclipse, which IBM developed and turned over in 2001 to the nonprofit Eclipse Foundation (www.eclipse.org) to manage as an open-source platform.

In addition to providing an IDE, Eclipse automates numerous functions that developers would otherwise hand code, said Alan Zeichick, editor in chief of *SD Times*, a newspaper for software-development managers.

Eclipse has garnered so much support that many industry observers say it is now the key Java-tools player. Today, the Eclipse Foundation has 98 member companies, including most of the largest software vendors. The technology even has its own annual conference, EclipseCon, which sold out this year.



"Eclipse has truly won," said Zeichick. It is inexpensive to use and makes it much easier to integrate their tools with one another, he added.

ECLIPSE HISTORY

Object Technology International developed the Java-based technology behind Eclipse before IBM bought the company in 1996. IBM began working on Eclipse internally in 1998 to integrate its many development programs.

IBM designed the Eclipse platform in accordance with standards set by the Object Management Group (www.omg.org), which produces and maintains specifications for interoperable enterprise applications.

Although the Eclipse Foundation now manages the platform, nonmembers can also build applications using the technology.

HOW ECLIPSE WORKS

Multivendor IDEs are a key factor in software design. They let a project's developers select their preferred tools from different vendors without worrying about making them work together or learning multiple interfaces and programming environments.

Like other IDEs, Eclipse is a programming environment packaged as an application. It consists of a code editor, compiler, debugger, GUI builder, and other tools.

For example, the Eclipse Foundation has included refactoring tools, which conduct a series of small transformations to restructure an existing body of code—for example, to make it smaller and less buggy—without changing its external behavior, noted Ian Skerrett, the organization's director of marketing.

The foundation has also added intelligence to the text editor, which is used for hand coding, Skerrett added.

Eclipse offers a set of APIs that connect tools into one unit, the Generic Workbench, that works as a single development environment with one set of behaviors and interfaces.

Eclipse uses the Standard Widget Toolkit to provide programs' interfaces. The IBM-created SWT is a class library for creating GUIs in Java. It lets developers build portable applications that directly access the user-interface facilities of the operating systems on which they are implemented. The Java programs thus look like native desktop applications.

Proponents say that because the SWT works with the operating system, it will perform better than techniques that bring their own UI features and thereby create user interfaces that look the same regardless of the host OS.

Meanwhile, Eclipse automates functions, such as the creation of buttons and dialog boxes, that developers might otherwise have to hand code.

Eclipse is built with Java and thus runs on multiple platforms. However, it will also help build applications in other languages such as C, C++, Cobol, and HTML.

REASONS FOR ECLIPSE'S SUCCESS

"Among the top Java IDEs, Eclipse is the only one gaining market share in

Published by the IEEE Computer Society

Europe, the Middle East, Africa, Asia-Pacific, and North America," said analyst Albion Butters with Evans Data, a market research firm.

"The power of Eclipse is the common platform that you can integrate different tools into," said the Eclipse Foundation's Skerrett.

Eclipse was created as a platform for plug-in tools that extend the IDE's capabilities so that it can work with numerous programming languages and applications, as Figure 1 shows. Anyone can write plug-ins for Eclipse and have them work directly with any other plug-ins for the platform. Some other IDEs limit plug-in creation to company partners.

There is thus a "huge number" of interoperable third-party plug-ins, which has made Eclipse very popular, said John Andrews, Evans Data's chief operating officer.

IBM's release of Eclipse to the Eclipse Foundation made the technology independent of any company, which fueled its broader adoption by businesses that don't want to be tied to a specific vendor, noted Rob Cheng, Borland's director of product marketing. "The more independent Eclipse is, the more comfortable companies and developers feel using it," he explained.

For example, Oracle is working to ensure that any developer using Eclipse can build applications for the vendor's application server and database, explained Ted Farrell, chief architect in the company's Application Development Tools Division.

Lower costs

The entire Eclipse development platform is free. Proprietary IDE systems such as JBuilder, JDeveloper, and JetBrains' IntelliJ IDEA, on the other hand, can cost up to $3,500 each.

Users seeking to add plug-ins that aren't part of Eclipse can get some tools for free and pay for others. Either way, it can be less expensive than buying an entire proprietary development platform.

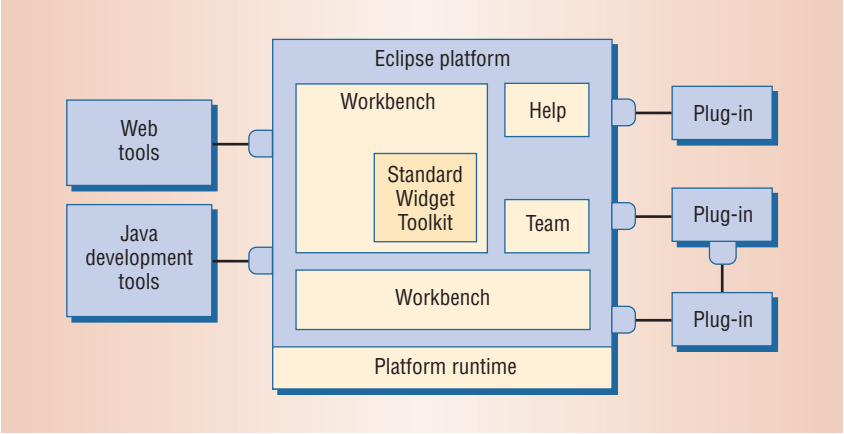


Figure 1. Eclipse was created as a platform for plug-in tools that extend its capabilities so that it can work with numerous programming languages and applications. The tools plugged into the platform operate on regular files in the user's workspace. Eclipse can place a project in the workspace under version and configuration management with an associated team repository. On startup, the platform runtime discovers the set of available plug-ins, reads their manifest files, and builds an in-memory plug-in registry. The workbench provides Eclipse's user-interface personality and includes the SWT general-purpose UI toolkit.

Eclipse's popularity has led many tool developers to make their products compatible with it. This competition has reduced the price of the plug-ins that aren't free, according to Evans Data's Andrews.

Fast-moving innovation and development

Because Eclipse is open source, Borland's Cheng said, developers have ready access to the source code and can modify it and innovate quickly to meet users' needs.

And, Cheng added, companies like the technology's open development process. "It is a very transparent process. Most of the communications, milestones and plans are public, and the builds are available for public download. Interim builds come out every couple weeks or every month so that people can try it out and give feedback quickly. There is a lot of community involvement," he said.

Elegant architecture

According to Cheng, Eclipse is a small, modular IDE with an elegant architecture that starts from a basic but powerful foundation. "There is a layer that lets you integrate applications without worrying about drawing dialog boxes, buttons, and widgets or property pages and project trees," he explained.

Thus, he elaborated, developers can hand code the new elements they need or want and disregard the elements that stay the same from program to program, such as dialog boxes.

NOT A TOTAL ECLIPSE

Although it appears to be the Java IDE of choice, Eclipse still faces competition from alternatives such as JBuilder, Visual J#, JDeveloper, and NetBeans.

"Eclipse is certainly a very popular IDE and very successful," said Tim Cramer, software engineering director for Sun's NetBeans. "However, NetBeans is seeing a resurgence with the developer community. Eclipse has been great competition, and because of this, we're all going to improve."

"With NetBeans 4.1," he added, "we now have a number of features that add value above and beyond what Eclipse might have: visual development of

J2ME (Java 2 Platform, Micro Edition) applications, debugging on a live cell phone, and an advanced profiler."

Thus, he said, "Our active users have gone up dramatically according to our internal measures, and we're also seeing a surge in traffic to our Web site."

Concerns

According to Cramer, because Eclipse and the SWT are not going through Sun's Java Community Process for introducing new features, they don't create applications with true Java functionality.

In addition, he explained, developers must port the SWT to all platforms on which Eclipse runs, which can be complex, time consuming, and expensive. NetBeans, on the other hand, runs natively anywhere there's a 1.4 or later version of Java, he noted.

NetBeans originally used the Abstract Window Toolkit, an API for Java-application GUI development. Sun discovered performance and extensibility limitations with AWT and thus developed Swing, explained Cramer.

AWT uses the operating system's graphics code for GUIs while Swing brings its own, creating GUIs that look the same on any OS. In addition, Cramer said, Swing has about 500 classes of GUI-related objects and thus offers richer graphics and more components than AWT, which has only about 50 classes.

Because AWT and Swing are part of the Java specification, proponents say, they offer better Java functionality.

Oracle's compromise

Concerned that Eclipse and NetBeans might create incompatible technologies that would split Java and make it less attractive to developers, Oracle has offered a compromise designed to enable compatibility.

The company has submitted Java Specification Request 198, "A Standard Extension API for Integrated Development Environments," to Sun's JCP. Rather than introduce yet another IDE, JSR 198 would provide a standard API that would work with all Java IDEs that support it.

The Eclipse Foundation has submitted for review Eclipse 3.1 Release Candidate 1, which features an updated SWT that offers more capabilities and interoperability with a greater number of browsers. The new version would also be faster, include more wizards, and enable automatic coding of additional features.

In addition, the foundation is expanding its activities. For example, the group's Web Tools Platform Project plans to begin releasing tools this summer. The organization has also developed business intelligence and reporting tools for generating reports from Java servers and is working on a rich-client platform for developing robust desktop and workstation applications.

According to *SD Times*' Zeichick, Eclipse will be the leading Java IDE for at least five years because of vendor support.

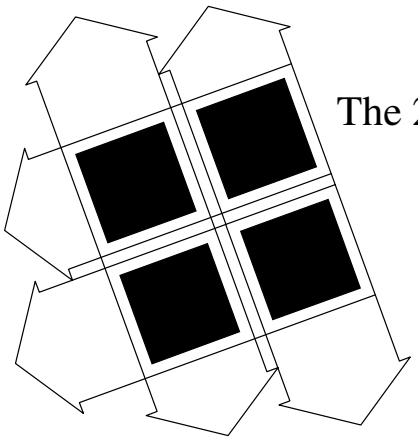
However, Cheng noted, it remains to be seen how sophisticated Eclipse's functionality and features will get. "It's not clear where different groups within Eclipse will move and evolve with their projects. It may be that Java development will only reach a certain level on Eclipse," he explained.

Said Oracle's Farrell, "Eclipse's success is tied to how good a product it is. If it starts to deviate from the main development base, it will begin to lose favor. Now that Eclipse is expanding, there are a lot more people contributing different types of technologies to it. As the base starts to grow, there is a danger of it losing some of its appeal as being lightweight, fast, and focused on the developer." ■

David Geer is a freelance technology writer based in Ashtabula, Ohio. Contact him at david@geercom.com.

Editor: Lee Garber, *Computer*, l.garber@computer.org

CALL FOR PAPERS



ICDCS 2006

The 26th International Conference on

Distributed Computing Systems

Lisbon, Portugal
July 4-7, 2006

http://icdcs2006.di.fc.ul.pt

http://www.computer.org

Sponsored by

The IEEE Computer Society Technical Committee on Distributed Processing

SCOPE

The conference provides a forum for engineers and scientists in academia, industry and government to present their latest research findings in any aspects of distributed and parallel computing. Topics of particular interest include, but are not limited to:

- •Algorithms and Theory
- •Autonomic Computing
- •Data Management
- •Fault-Tolerance and Dependability
- •Internet Computing and Applications
- •Network Protocols
- •Operating Systems and Middleware
- •Parallel, Cluster and GRID Computing
- •Peer-to-Peer
- •Security
- •Sensor Networks and Ubiquitous Computing
- •Wireless and Mobile Computing

WORKSHOPS

Workshops will be held in conjunction with the conference. Workshop proposals should be submitted to Workshops Co-Chair Ricardo Jiménez-Peris (rjimenez@fi.upm.es) by **July 30, 2005**. Please see the conference web page for details.

PAPER SUBMISSION

Form of Manuscript: Not to exceed 25 double-spaced, 8.5 x 11-inch pages (including figures, tables and references) in 10-12 point font. Number each page. Include an abstract, five to ten keywords, the technical area(s) most relevant to your paper, and the corresponding author's e-mail address.

Electronic Submission: Submissions will be handled via the conference web page.

The proceedings of the conference and the workshops will be published by the IEEE Computer Society Press.

IMPORTANT DEADLINES

Paper Submission	**November 15, 2005**
Author Notification	February 15, 2006
Final Manuscript Due	March 29, 2006

For Further Information, Please Contact: ler@di.fc.ul.pt

ORGANIZING & PROGRAM COMMITTEES

General Co-Chairs
Haruhisa Ichikawa
NTT Network Innovation Labs, Japan
Michel Raynal
Université de Rennes, France

Program Co-Chairs
Mustaque Ahamad
Georgia Institute of Technology, USA
Luís Rodrigues
Universidade de Lisboa, Portugal

Program Vice Chairs

Algorithms and Theory
Philippas Tsigas
Chalmers, Sweden

Autonomic Computing
Manish Parashar
Rutgers, The State U. of New Jersey, USA

Data Management
Karl Aberer
EPFL, Switzerland

Fault-Tolerance and Dependability
Eliane Martins
UNICAMP, Brazil

Internet Computing and Applications
Ling Liu
Georgia Institute of Techology, USA

Network Protocols
Katherine Guo
Bell Labs, USA

Operating Systems and Middleware
Roy Friedman
Technion, Israel

Parallel, Cluster and GRID Computing
Tarek S. Abdelrahman
University of Toronto, Canada

Peer-to-Peer
Bobby Bhattacharjee
University of Maryland, USA

Security
Christian Cachin
IBM Zurich Research Lab, Switzerland

Sensor Networks and Ubiquitous Computing
Tarek Abdelzaher
University of Virginia, USA

Wireless and Mobile Computing
Arup Acharya
IBM Research, USA

Program Committee
(Please see the web page for list)

Workshop Co-Chairs
Makoto Takizawa
Tokyo Denki University, Japan
Ricardo Jiménez-Peris
Universidad Politécnica de Madrid, Spain

International Liaison Chair
Ten H. Lai
The Ohio State University, USA

Award Co-Chairs
Anish Arora
The Ohio State University, USA
Joseph E. Urban
Arizona State University, USA

Treasurer
Filipe Araújo
U. Lisboa, Portugal

Local Arrangements Chair
José Rufino
U. Lisboa, Portugal

Publicity Chair
António Casimiro
U. Lisboa, Portugal

TCDP Chair
Chita Das
Penn State University, USA

Steering Committee Chair
Ming T. (Mike) Liu
The Ohio State University, USA

Instant Messaging: A New Target for Hackers

Neal Leavitt

Instant messaging is exploding as a means of personal and corporate communications. Individuals chat via IM; companies rely on beefed-up versions of the technology, with its real-time capabilities, for collaborative design work; and e-businesses use IM to provide live, immediate customer service to shoppers.

Market research firm IDC estimates that by 2008, more than 506 million people worldwide will use an IM product. The Radicati Group, another market research company, predicts that there will be 78 million enterprise IM users by the end of 2008.

Meanwhile, the technology is finding its way onto mobile devices, including PDAs and smart phones.

However, as IM becomes more popular, particularly for businesses, it has also increasingly become the target of attacks, such as those using malicious code and phishing.

"Over the past several months, IM viruses and worms have grown an astronomical 1,600 percent compared to last year," said Jon Sakoda, chief technology officer for IM software vendor IMlogic.

Attacks against major IM networks rose almost 400 percent from five during the first quarter of 2004 to 24 during the same time period this year, according to IM security vendor Akonix Systems.

Some security experts say IM is following the same patterns shown during the development of e-mail attacks. These include the use of tricks to encourage victims to click on virus-laden attachments or hyperlinks to Web pages that upload applets to either infect visitors with malware or drop unwanted software on their computers.

"Most of this," Sakoda said, "is relatively benign adware or spyware, but there have been several IM worms that have attempted to shut down security software and disable system applications."

The most dangerous part about the attacks is their speed of propagation, caused by IM's real-time capabilities, he noted. According to Eric Chien, principal software engineer for antivirus vendor Symantec, the company ran a simulation in late 2004 that showed IM viruses could spread to 500,000 machines in less than 30 seconds.

Traditional antivirus technology, in which vendors typically need 24 hours to find remedies for new malicious code, may be too slow to prevent many IM attacks from spreading rapidly.

GROWING PROBLEM

The IMlogic Threat Center (http://imlogic.com/im_threat_center/index.asp)—a consortium of security and IM providers such as AOL, McAfee, Microsoft, Symantec, and Yahoo—said 82 percent of IM attacks included virus or worm propagation.

According to the center, 64 percent of attacks targeted Microsoft's widely used systems, particularly MSN Messenger, 11 percent hit Yahoo Messenger, and 25 percent affected AOL's AIM and ICQ systems.

In June, noted IMlogic's Sakoda, hackers began shifting focus to AOL, but MSN is still a favorite because of the Microsoft connection and the widespread distribution of the Windows messenger client on PCs.

Also, explained Symantec's Chien, "Microsoft provides a well-documented API for MSN Messenger that allows one to control it and thus send out worms via IM."

"For virus authors who want their 15 minutes of fame or criminal organizations that want the largest cash cow, then Microsoft is the biggest animal to run down," said Jamie Lyndon Yaneza, senior antivirus research consultant for TrendLabs, a subsidiary of antivirus company TrendMicro.

Driving forces

Hackers have the same motivations—such as financial gain, enhancing their reputation among peers, solving a technical challenge, and creating mischief—for attacking IM systems as they do for targeting e-mail or other network-based technologies.

However, e-mail has been a more attractive target than IM for many years. Popular public IM systems such as AIM and Yahoo Messenger are closed and thus don't generally connect to other systems. This limits IM's ability to spread attacks. Also, until recently, IM clients have been simple

Published by the IEEE Computer Society

systems with few published vulnerabilities to exploit.

In addition, IM protocols are proprietary, which has made them more difficult to reverse engineer, explained Ero Carrera, a researcher at antivirus-software vendor F-Secure. E-mail, on the other hand, uses publicly available standards such as the Simple Mail Transfer Protocol (SMTP), he noted.

However, after years of e-mail attacks, users and security firms have shored up their defenses. Hackers have thus turned their attention to IM, said TrendLabs' Yaneza.

He added that IM's informality and immediacy causes many users to let their guard down when using the technology, something that is not the case with e-mail, whose risks are better known.

And adolescents, who comprise the fastest-growing segment of IM users, don't generally practice safe computing as much as adults, said Craig Schmugar, virus research manager for antivirus company McAfee.

Meanwhile, as IM's functionality has increased, systems have become more complex and vulnerabilities have crept in.

IM vulnerabilities

As a messaging system, IM suffers from many of the same vulnerabilities as e-mail. For example, IM users can launch a hacker's attack by inadvertently opening infected attachments.

Users can also click on a hyperlink in an instant message that leads them to a phisher's counterfeit bank or e-commerce Web site. The site asks them to enter their user name, password, bank account and Social Security numbers, and other personal information that hackers can subsequently sell or use illegally.

In addition, IM supports the peer-to-peer transfer of files and messages with attachments, so they bypass most of e-mail's server- and security-gateway-based virus scanning.

Password protection is limited in most IM systems, and the communications are rarely encrypted. "Without encryption, any off-the-shelf sniffer can reveal the content of IM communications," said Marcus Sachs, a computer scientist at SRI International, a contract research institute, and deputy director of the US Department of Homeland Security's Cyber Security R&D Center.

As IM has grown more popular, it has become the target of attacks.

Unlike e-mail, which usually uses SMTP and TCP/IP port 25, IM systems use various ports and proprietary protocols. For example, AIM and ICQ use port 5190, MSN Messenger uses port 1863, and Yahoo Messenger uses ports 80 and 5050. This lack of consistency makes it difficult for IT departments to monitor IM communications for attacks and threats.

No corporate IM policies

IM problems are caused not only by common coding mistakes but also by a lack of corporate IM-use policies. A survey of US businesses by SurfControl, a corporate Internet security vendor, found that 90 percent of respondents had an Internet-access policy but only 51 percent had an IM policy.

Many companies don't recognize IM's dangers, noted Tim Johnson, director of the IMlogic Threat Center. And many organizations that don't use IM for corporate communications aren't aware that employees are using the technology on their own, as they can frequently download popular IM systems from the Web themselves.

IM ATTACKS

IM attacks are like those that affect e-mail and other types of network-based assaults.

Malicious code

IM attacks have included various types of Trojan horses and worms.

Assiral.A. This simple mass-mailing worm arrives as a Windows 32-bit executable that deletes files and modifies Internet Explorer homepage settings.

Bizex. The main component of this worm, which attacks ICQ systems, has spying and data-stealing capabilities. Bizex spreads by sending a hyperlink to a victim's contacts. Clicking on the link sends them to a Web page that uploads the worm.

Bropia. This worm and its variants, including Kelvir and Serflog, spread via MSN Messenger. They copy themselves into a Windows system directory, download more malware onto the victim's computer, and reduce system security. Some variants hide on a PC, only to re-emerge at a later date.

Buddypicture. The attack by this Trojan, which affects AIM systems, starts with an instant message that includes a hyperlink to a Web site supposedly featuring pictures of the purported sender, whose name was on the victim's contact list. The message asks the victim to download an applet first. If downloaded, the applet uploads adware and spyware to victims' computers.

Gabby.a. The Gabby worm attacks AOL's AIM and ICQ systems by sending recipients a hyperlink and tricking them into clicking on it. Victims then get to a Web page that uploads spyware, as well as a worm that opens a backdoor to the machine and eliminates Windows services such as those used with antivirus and firewall software.

Kelvir. This worm spreads by sending a hyperlink to MSN Messenger users with messages such as "Hey, check this out" or "LOL, this is a funny picture of me." Users who click on the link go to a Web page that uploads the virus to their computers. Kelvir then spreads via victims' buddy lists.

The worm can turn computers into spam broadcasters, log keystrokes such as those in user names and passwords, and e-mail the information to hackers.

Kelvir recently shut down international media company Reuters' pro-

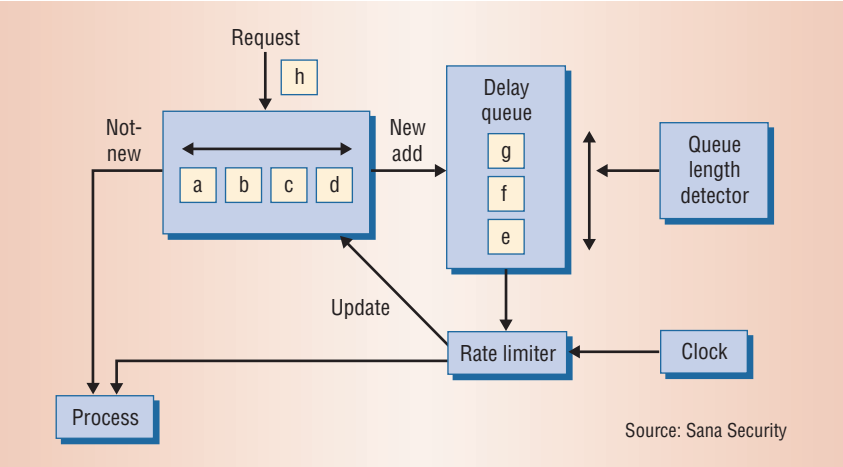


Figure 1. When a security system spots worm-like behavior on an IM network, virus throttling slows the spread of the malware and thus limits the damage. The technique compares a new connection that an IM client is trying to make—in this case to h—to a short list of frequently made, and thus presumably safe, connections—in this case a, b, c, and d. If the new connection is on the list, the system lets it pass. If it is a new connection, the system places it on a delay queue, which in this case already holds messages to e, f, and g. If there is a lot of traffic to many different destinations, as occurs with a virus, the delay queue gets large and the system stops further transmission.

prietary, closed, 6,000-user IM system, which is based on Microsoft technology.

Phishing

IM phishing is an industry-wide issue. For example, phishers recently attacked Yahoo Messenger by sending a message containing a hyperlink to a counterfeit Yahoo Web site. The site displayed a sign-in screen and asked victims to log in with their user ID and password. With this information, an attacker could sign in to the victims' Yahoo Messenger accounts and hack into their contact lists and user profiles, which can contain personal and financial information.

According to Yahoo Messenger director Frazier Miller, the company has enhanced security by adding a new SpamGuard feature that lets consumers report spam or unsolicited IM messages. In addition, it blocks communications from previous senders of unsolicited messages. The company also started the Yahoo Security Center (http://security.yahoo.com), which teaches consumers how to protect themselves online.

Hijacking

IM worms can let an attacker hijack and send messages with infected attachments or phishing-related hyperlinks from victims' clients to their IM contacts.

This could make the contacts believe the communications came from an acquaintance and that opening attachments or clicking on hyperlinks is safe.

Denial-of-service attacks

An attacker could launch a DoS attack by sending many specially crafted TCP/IP packets to servers in an IM provider's infrastructure and thereby prevent legitimate messages from passing through.

Hackers could also send many packets to an IM user to hang up or crash the messaging client or eat up CPU resources and destabilize the computer.

ADDRESSING THE THREATS

Messaging providers and security companies are taking steps to combat IM attacks, such as establishing the IMlogic Threat Center, which monitors and analyzes IM security risks, warns users against vulnerabilities, and provides threat management. Its members include about 25 companies, which fund the organization, and about 400 individuals.

IM providers and security companies also advocate educating consumers about safe computing practices.

Upgrading IM technology

IM attacks can cause buffer overflows, which occur when a program or process tries to store more data in a buffer than it was designed to hold. The extra information overflows into adjacent buffers, corrupting or overwriting valid data. The overflowing data can contain instructions designed to cause problems such as client failure or the consumption of CPU or memory resources.

Poor programming and memory management can enable buffer overflow attacks. Thus, major IM networks are revising their clients to ensure better memory management.

Sana Security's Primary Response protects against buffer overflows by preventing the type of code execution that occurs during the attacks.

Primary Response also includes a profile of normal file and network activity so that the system can detect anomalous behavior that indicates an IM-based or other attack. The product also includes Sana's Active Malware Defense Technology, which recognizes programs behaving maliciously.

Firewall maker Zone Labs makes IMSecure, which can detect viruses; block spam, IM-borne scripts, and buffer overflow attacks; and encrypt data being sent to another IMSecure user. Users can also choose to block certain IM features, such as file transfers.

Symantec and McAfee added IM scanning and the ability to remove malware from attached files to their Norton AntiVirus and VirusScan products, respectively. And TrendMicro's InterScan Web Security Suite filters Web traffic for the URLs of Web sites known to be involved in malicious downloads, phishing, and spam.

To limit the damage that infected files can cause, Microsoft has designed MSN Messenger so that it won't transfer several types of files, such as executables, command files, and program information files (which tell Windows how to run non-Windows applications).

Meanwhile, vendors are starting to release end-to-end encryption plug-ins for IM clients.

IM-use policies

"Companies need to have a policy on IM, even if it's to ban it," said SRI International's Sachs. "The best policy is to provide for a way that employees can use IM safely and describe how the technology will be used [only] to support business needs."

According to SurfControl, IM-security policies could limit which users can access IM networks; route instant messages through the secure enterprise network; require regularly updated, real-time message-content filtering; mandate virus scanning of all file transfers; and block transmission of hyperlinks over IM.

Slowing IM worms' spread

Traditional antivirus technology reacts too slowly to stop many IM virus outbreaks. *Virus throttling*, a promising alternative that is still experimental for IM, slows the spread of messaging worms and thus limits their damage, rather than prevent the infections, as Figure 1 shows.

When a system spots wormlike behavior on an IM network, virus throttling limits the number of IM messages an infected user can send outside the small group of contacts with which they communicate most frequently, explained Matthew Williamson, a Sana Security senior research scientist who developed the technique while at Hewlett-Packard.

Said Trend Labs researcher Ivan M. Macalintal, "Attacks will increase in sophistication." For example, IM malicious code will make itself harder to detect by mutating several of the elements that security systems use to identify it. For example, the malware may mutate the code itself to defeat the code signatures that antivirus software uses to detect malware, noted the IMlogic Threat Center's Johnson.

And in the near future, said F-Secure's Carrera, wireless-IM security problems might arise.

IM's rapid growth in the enterprise and lack of deployed IM security technology continue to make it attractive to attackers. "IM has become an infection vector alternative to e-mail, and we will see a gradual increase of threats simply because of the bulk of users," said Jim Murphy, SurfControl's director of product marketing.

According to Murphy, large organizations will be slow to react to the threat but eventually will be compelled to do so by the risks involved.

Neal Leavitt *is president of Leavitt Communications, an international marketing communications company based in Fallbrook, California. He writes frequently on technology-related topics. Contact him at neal@leavcom.com.*

Editor: Lee Garber, *Computer*, l.garber@computer.org

Putting a Business Suit on Grid Technology

Two initiatives hope to make grid computing—in which scattered computers are linked together to function as a single machine—more useful to businesses. Academia has used grid computing for many years.

One focus of the new efforts is to create industry standards, which experts believe are necessary to widen corporate adoption of grid technology. Another focus is explaining the details and benefits of efficiently implementing grid technology.

The Enterprise Grid Alliance (www.gridalliance.org), a group of computing companies, plans to release white papers on how to make grids more practical for corporate use.

The EGA has also developed a common set of terms that all involved parties can use to help with the development of standards and the implementation of corporate grid systems.

The EGA is working closely with standards-related, grid-oriented, and other organizations, such as the Global Grid Forum. The alliance will rely on these organizations to create grid-computing specifications, said Paul Strong, chair of the EGA technical steering committee and a Sun Microsystems systems architect.

Today, most of the few companies doing grid computing use vendor-specific tools from companies such as Data Synapse, IBM, Oracle, and United Devices, said Jonathan Eunice, an analyst at Illuminata, a market research firm. Standards would help widen grid adoption by allowing system interoperability and providing common development criteria.

The Globus Alliance (www.globus.org), a consortium of grid researchers and research institutions, has released the open source Globus Toolkit 4.0 for writing applications that run on grid systems.

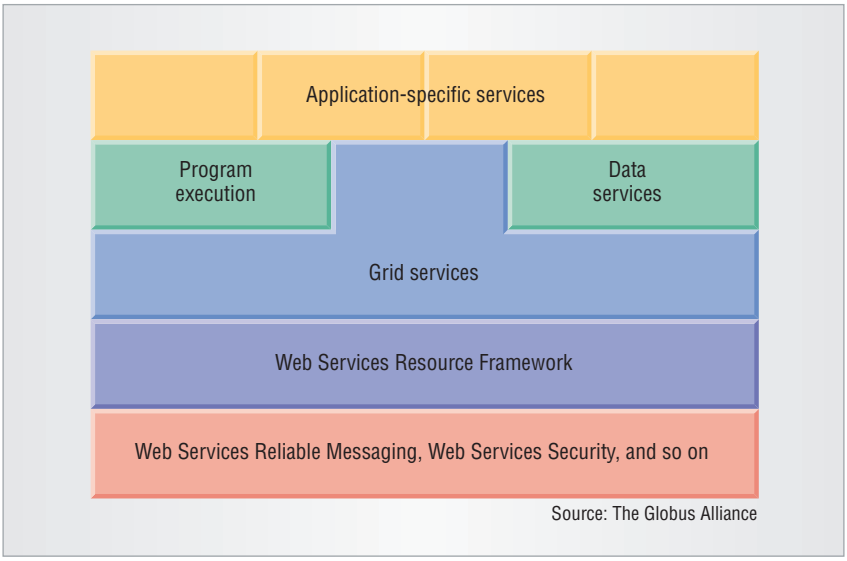
The toolkit manages distributed-computing and -storage resources. Companies could then build high-level enterprise grid-based applications atop the toolkit, explained Ian Foster, senior scientist and head of Argonne National Laboratory's Distributed Systems Lab and a member of the Globus Alliance's board of directors.

To conduct distributed computing on heterogeneous systems, Foster noted, the toolkit uses standardized interoperability technologies such as Web services and Grid FTP.

By letting companies distribute work among a system of computers, grid technology offers businesses versatility, agility, and improved efficiency, Strong said.

Grids can also help companies save money by performing tasks via the unused time of existing computers, rather than by buying new servers, he added. If a machine isn't used for a certain amount of time, the grid server can offload a job to it. When the workstation is used again, the system moves the task to an available computer.

Companies could use grid computing for problems that can be divided into pieces for assignment to multiple PCs, such as complex analytics, computation-heavy activities, and engineering applications with large or bursty workloads, explained Foster. ■



To conduct distributed computing on heterogeneous systems, grid-based applications use standardized interoperability technologies such as Web services. This approach features grid and application-specific services built on the WS-Resource Framework, which defines a family of specifications for accessing resources using Web services. They are supported by lower-level protocols such as WS-Security, used to provide secure Web services-based communications, and WS-Reliable Messaging, which enables dependable message delivery between distributed applications.

Unusual Attack Holds Computer Files for Ransom

Hackers have launched a new type of attack in which they remotely lock up documents on computers and demand a ransom from the victims to unlock them. Although this attack wasn't widespread, experts fear criminals could up the stakes.

Security researchers at Websense, a Web-filtering- and Web-security-software vendor, discovered the *ransomware* scheme when hackers victimized a corporate customer's computer and left a ransom note. The company declined to provide specifics about the victim or many details of the attack.

The attack infected computers when users visited a Web site vandalized by hackers who added malicious software to it by exploiting a Web-server or operating-system vulnerability.

Hackers then took advantage of a problem with the Internet Explorer help function that let them upload software to a victim's computer and then run it. Microsoft provided a patch for this problem last summer.

The attack used the vulnerability to infect victims' unpatched PCs with the Trojan.Pgpcoder Trojan horse. Once downloaded and run, this program downloads a second application that searches for and scrambles 15 file types, including word-processing documents, digital photographs, and spreadsheets.

The attack leaves a ransom note in the only readable text file left on the infected computer. According to Hubbard, the note contains instruc-

Cornell Scientist Builds Self-Replicating Robot

Cornell University researchers have developed small, simple robots that can build copies of themselves.

Project leader and assistant professor Hod Lipson has created a family of robots made up of three or four cubes, each with four-inch sides. The cubes are cut diagonally into halves that can rotate against each other, thereby letting the cubes connect to one another in different ways. This lets robots change shape.

Lipson said his project demonstrates important concepts that could be used to develop machines that repair themselves, creating robots that are self-sustaining as well as durable.

The process could lead to self-repairing robots for use in circumstances in which human intervention isn't practical or possible, such as remote exploration, space flights, or hazardous situations.

Each cube is identically equipped and includes a microprocessor with information from the parent robot on the original machine's body shape and the cube's designated position in the structure. The cube also contains replication instructions, an electrical power and data transmission contact on its face, and a gear box. Depending on its position in the robot's blueprint, each cube executes a different part of the program.

The robots, powered by electrical contacts on a table's surface, self-replicate by using other cubes placed in "feeding" locations. The cubes connect to one another via magnets on their faces. The parent robot picks up the first few cubes in the replication process. The child robot picks up some of the later cubes and configures itself properly.

A four-cube robot built a replica in 2.5 minutes, Lipson said.

Lipson noted that his research team—which includes students Bryant Adams, Efstathios Mytilinaios, and Viktor Zykov—is exploring the more complex process of developing self-repairing robots, which will require diagnostics programs and possibly the ability to build temporary scaffolds. ■

Researchers have developed simple robots, made of cubes, that can build copies of themselves. Each cube contains a microprocessor with information on the original machine's body shape, replication instructions, an electrical power and data transmission contact on its face, and a gear box. The robots receive power via electrical contacts on a table's surface.

tions to send a message with a specific subject line to a designated e-mail address.

A reply then tells the victim to send hackers $200 via e-gold—an e-payment company—in return for the unscrambled files. Hubbard said the attackers probably asked for only $200 to encourage payment.

The attack used apparently is a form of *weak obfuscation*, which entails file scrambling, said Dave Cole, director of security product management for Symantec Security Response. The technique was sufficient to make it difficult for victims to get their files without either paying the attackers or obtaining expert assistance, he noted.

Investigators could follow the money trail, say security experts. However, Cole said, because law-enforcement agencies have limited resources and because no one has reported giving money to the attackers, police might not investigate the case unless it becomes more significant.

Experts said there were no reports that the new threat was spreading. In addition, the vandalized Web sites that spread the infection are no longer active. However, the problem is still significant because attackers could use e-mail or other means besides infected Web sites to distribute Trojan. Pgpcoder.

According to Cole, "This appears to be a proof-of-concept attack."

Security researchers worry that improved versions of the attack might be more dangerous and more difficult to either prevent or solve. ■

Schools Increasingly Use Software to Grade Essays

Schools are increasingly turning to software to analyze and grade student essays. Computers have long graded multiple-choice tests, which entail the easily automated task of matching a student's response to the correct answer. However, scoring essays is a more complex task involving a range of variables and requiring subtle analysis.

Software now scores a variety of written assignments, from high school papers to an essay that appears on the Graduate Management Admission Test (GMAT), the standard exam for US graduate business-school applicants.

Most essay-grading software analyzes sentences and paragraphs, looking for keywords and relationships between terms that indicate the writer has properly explained important concepts.

The systems use a number of technologies to determine how well an essay has met an assignment's requirements. These technologies include natural language understanding to recognize keywords and word patterns, case-based reasoning to help compare unanalyzed essay segments with analyzed segments, and machine learning to identify patterns that could help with the evaluation process.

Typically, these systems extract information from a pool of human-graded essays to develop their own comparable evaluation approaches.

e-Rater essay-grading software by ETS, a major educational-assessment company that offers the GMAT and other tests, can develop its grading techniques for an assignment by analyzing papers that aren't even necessarily on the assigned theme, explained Jill Burstein, ETS's principal development scientist.

"Most programs use statistical modeling to build a predictive model," explained Ed Brent, a University of Missouri sociology professor who designed SAGrader (pronounced "essay grader") software, which he uses to grade first drafts of papers in his introductory sociology classes.

SAGrader uses modeling to come up with the material that should appear in essays that properly address the assigned topic, including definitions of terms and supportive information, he said. The application also generates comments to students.

Frequently, to make essay-grading software work properly, teachers must prepare it for specific assignments, such as by entering keywords and important elements, taking into accounts different ways students might express them.

Proponents say essay-grading software reduces some of the subjectivity and tedious work that occurs when teachers grade papers. It also reduces grading-related overtime costs, noted Stan Jones, Indiana's commissioner of higher education. Indiana high schools are using software to grade year-end English assessments for 60,000 juniors. Jones noted that the machines tend to grade the essays "marginally higher" than teachers.

Skeptics have said that using software to grade essays would encourage students to figure out ways to trick the technology, such as by scattering obvious keywords or phrases meaninglessly throughout a paper. ■

News Briefs written by ***Linda Dailey Paulson****, a freelance technology writer based in Ventura, California. Contact her at ldpaulson@yahoo.com.*

Editor: Lee Garber, ***Computer****;*
l.garber@computer.org

INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

CALL FOR PAPERS

- 35th Annual Conference -

2006 International Conference on Parallel Processing (ICPP 2006)

http://www.cse.ohio-state.edu/~icpp2006

Columbus, Ohio, USA
August 14-18, 2006

Sponsored by
The International Association for Computers and Communications (IACC)

In cooperation with
The Ohio State University, USA

35th Anniversary

The International Conference on Parallel Processing is celebrating its 35th year. To commemorate this event, a DVD will be issued containing all the proceedings of this and the previous 34 conferences.

Scope

The conference provides a forum for engineers and scientists in academia, industry and government to present their latest research findings in any aspects of parallel and distributed computing. Topics of interest include, but are not limited to:

- Architecture
- Cluster Computing
- Compilers and Languages
- Network Services
- Network-based / Grid Computing
- O.S. & Resource Management
- Parallel Algorithms and Applications
- Peer-to-Peer Technology
- Tools and Performance Analysis
- Wireless and Mobile Computing
- Systems Support for Parallel and Distributed Applications

Paper Submission

Paper submissions should be formatted according to the IEEE standard double-column format with a font size 10 pt or larger. Each paper is strictly limited to 8 pages in length. Submissions should represent original, substantive research results. We will not accept any paper which, at the time of submission, is under review for or has already been published (or accepted) for publication in another conference or journal venue. See the conference website for electronic paper submission instructions.

Conference Timeline

Paper Submisison Deadline	**February 1, 2006**
Author Notification	April 1, 2006
Final Manuscript Due	May 1, 2006

Workshops will be held on August 14 and 18. Workshop proposals should be submitted to the Workshops Co-Chairs, F. Ozguner (ozguner@ece.osu.edu) and T. Pinkston (tpink@charity.usc.edu) and by **August 1, 2005**.

Proceedings of the conference and workshops will be available at the conference and will be published by the IEEE Computer Society.

For Further Information - Please contact
Professor D.K. Panda, The Ohio State University, panda@cse.ohio-state.edu
Professor Wu-chi Feng, Portland State University, wuchi@cs.pdx.edu

Organizing & Program Committees

Honorary Chair
Tse-yun Feng, Penn State University, USA

General Chair
D.K. Panda, Ohio State University, USA

Program Chair
Wu-chi Feng, Portland State University, USA

Program Vice-Chairs

Architecture
John Carter, University of Utah, USA

Algorithms and Applications
David Bader, Georgia Inst. of Tech., USA

Cluster Computing
Daniel Katz, JPL/CalTech, USA

Compilers and Languages
Calvin Lin, University of Texas, USA

Network-Based/Grid Computing
Xiaodong Zhang, William and Mary, USA

Network Services
Yuanyuan Yang, Suny Stony Brook, USA

OS & Resource Management
Ron Brightwell, Sandia National Lab, USA

Peer-to-Peer Technology
Manish Parashar, Rutgers University, USA

Systems Support for Parallel and Distributed Applications
Chengke Wu, Xidian University, China

Tools and Performance Analysis
Darren Kerbyson, LANL, USA

Wireless & Mobile Computing
Kyongsook Lee, Univ. of Denver, USA

Program Committee Members
(Please see the conference web page.)

Workshops Co-Chairs
Fusun Ozguner, The Ohio State Univ., USA
Tim Pinkston, USC, USA

Awards Co-Chairs
Jose Duato, Univ. of Valencia, Spain
Wu-chun Feng, Los Alamos Nat'l Lab, USA

Publication Co-Chairs
Dong Xuan, The Ohio State Univ. , USA
Wei Zhao, Texas A & M, USA

Publicity Co-Chairs
Mohammed Banikazemi, IBM, USA
Nectarios Koziris, NTUA, Greece

International Liaison Co-Chairs
Steve Lai, The Ohio State Univ., USA
Makoto Takizawa, Tokyo Denki Univ., Japan

Local Arrangements Chair
Mario Lauria, The Ohio State Univ., USA

Registration Chair
Elizabeth O'Neill, The Ohio State Univ., USA

Steering Committee Co-Chair
Ming T. (Mike) Liu, The Ohio State Univ., USA

Securing Wi-Fi Networks

Hackers can decrypt and read data on a wireless link protected by built-in WEP encryption, and may even be able to access the data on a wired network through a Wi-Fi access point. The authors assess Wi-Fi network security in one city, analyze alternative security techniques, and suggest ways to secure such networks.

Kjell J. Hole
University of Bergen

Erlend Dyrnes
Ernst & Young—Bergen

Per Thorsheim
EDB Business Partner

Wi-Fi networks,[1] based on the IEEE 802.11b/g standards, have become very popular in recent years. Many users have installed Wi-Fi networks at home, and numerous corporations have added Wi-Fi access points to their wired networks, giving employees easier access to corporate data and services.

The scenario in which an employee connects to the corporate network from a home network is of particular interest. Although IT personnel control Wi-Fi access points in the corporate network, they cannot control, and are not necessarily even aware of, access points in home networks. These networks have thus given hackers new opportunities to gain unauthorized access to corporate computer systems and their data.

A review of the results of an investigation conducted to assess the security level in Wi-Fi networks in the city of Bergen, Norway, provides a context for analyzing some popular wireless security techniques and for offering suggestions on how to better protect these networks from hacking.

WIRELESS HACKING

Strictly speaking, a *hacker* is a software or hardware enthusiast who likes to explore the limits of programming code or computer hardware. However, the term more commonly refers to a person who breaks into or disrupts computer systems or networks to steal data or create havoc by uploading malicious code.

Wireless hackers specialize in Wi-Fi networks and employ a number of techniques to locate local area network nodes or *hotspots*. For example, *war-driving* involves driving through an inhabited area and mapping houses and businesses with Wi-Fi networks, usually using software on a wireless-enabled laptop.

War-walking, or walk-by hacking, involves walking through a neighborhood with a Wi-Fi-enabled personal digital assistant. PDA owners whose devices have a Wi-Fi client card can unintentionally war-walk if the operating system automatically connects the device to a Wi-Fi access point when the user passes by.

A war-walker with mischievous designs may engage in *war-chalking*—marking special symbols on sidewalks or walls to indicate the security status of nearby Wi-Fi access points. Our study indicated that war-chalking does not seem to be a widespread phenomenon in Bergen.

Wireless hackers pose a security threat because the encryption mechanism originally developed for Wi-Fi networks, known as Wired Equivalent Privacy, has been broken. In fact, it is possible to download programs to crack the encryption key on any WEP-encrypted link, as long as enough traffic is transmitted over the link. As the "Wireless Hacking Tools" sidebar illustrates, these programs are available for various platforms.

In addition, a number of books describe ways to attack Wi-Fi networks.[2-4] These books outline how to use different software tools to map wireless net-

Published by the IEEE Computer Society
0018-9162/05/$20.00 © 2005 IEEE

works, analyze the traffic on wireless links, crack WEP keys, and determine whether other security techniques have been implemented.

If WEP is the only encryption mechanism, wireless hackers can use one of the available cracker programs to decrypt the information. They can also obtain an IP address from the Wi-Fi network and gain Internet access to upload spam, viruses, worms, or Trojan horses or to download illegal material. Many freely available hacker tools also make it possible to access data on the wired network attached to the Wi-Fi access point.

WI-FI SECURITY IN BERGEN

Bergen is Norway's second largest city with 235,000 inhabitants. Before our investigation, we knew little about the security of Bergen's Wi-Fi networks or the threat from wireless hackers. However, based on earlier research in Oslo, the capital, we anticipated that there would be many such networks.

Some results

To assess the security risks, we engaged in both war-walking and war-driving in three areas of interest: the city center, which contains many shops and small businesses; Kokstad/Sandsli, an area close to the airport with large businesses; and Fyllingsdalen, a location outside the city center with many large office buildings. We used these tools only to collect research data; we did not reveal the exact locations of any discovered Wi-Fi networks, nor did we break any encryption.

We found no less than 706 wireless networks in Bergen. More than 500 were in the city center. Only 244 of the 706 networks used WEP. Of course, we cannot conclude that the remaining 462 transmit in the clear, but random spot checks strongly indicated that many networks in Bergen do not utilize any form of encryption.

Figure 1 depicts our war-driving results (including a few smaller areas not discussed here). We found that a wireless network's service set identity, as shown in the map, is often the name of the owner, a street address, or the name of the company owning the network. Of the 706 networks found, 166 had default names assigned by the manufacturer.

Implications

Due to their high complexity, inevitable bugs, emergent properties unanticipated by designers, and ever-changing technologies, few people appreciate the difficulty of securing computer networks.[5] In this context, Wi-Fi is just another new technology that makes it even harder to secure a large network. From a hacker's point of view, adding a wireless extension to a wired network could make it easier to access network resources.

Wireless Hacking Tools

The Internet is the perfect medium for distributing wireless hacking software. Some of these programs only list the names—known as service set identities—of the discovered networks, the channels they use, and whether or not WEP is active; other programs also crack WEP keys and support packet capturing as well as packet reinjection.

Wireless hacking tools are available for different platforms. Mac OS X tools for finding IEEE 802.11b/g wireless networks include KisMAC (http://kismac.com), which passively detects networks (promiscuous mode) and cracks WEP keys, and iStumbler (www.istumbler.net) and MacStumbler (www.macstumbler.com), both of which broadcast probe requests.

Linux and BSD tools include Kismet (www.kismetwireless.net), which provides passive network detection, and AirSnort (http://airsnort.shmoo.com), which passively detects networks as well as cracks WEP keys. NetStumbler (www.netstumbler.com) is a Microsoft Windows tool that broadcasts probe requests.

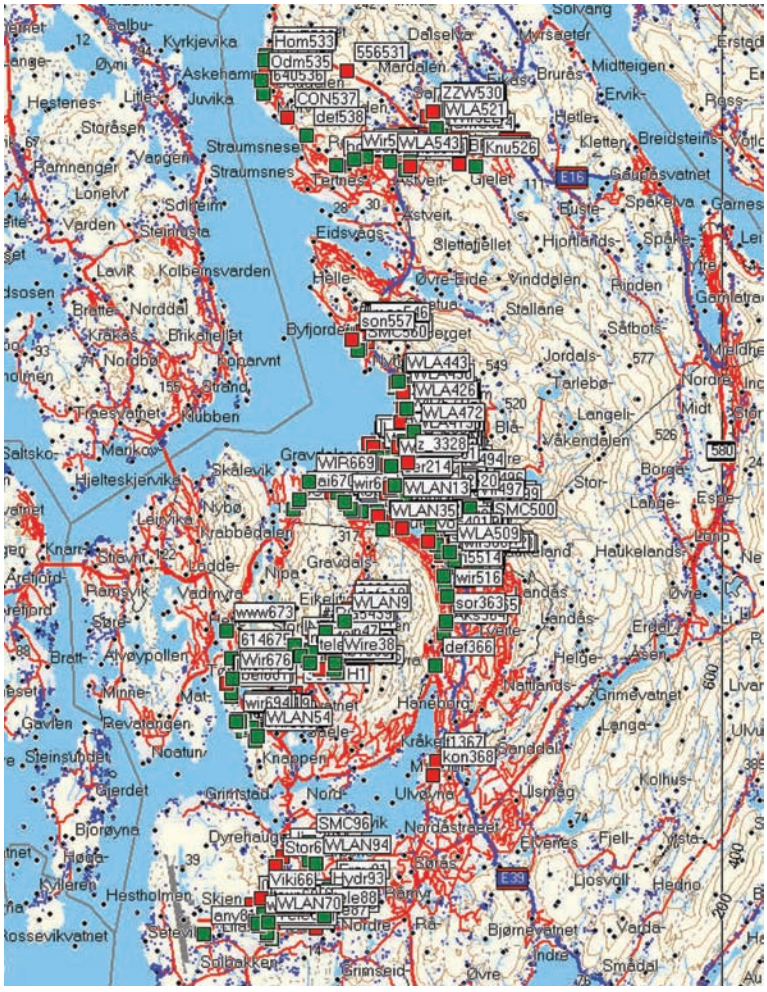


Figure 1. Wi-Fi networks in Bergen, Norway. Most of the 706 wireless networks revealed by war-driving did not use encryption.

Cracking two packet keys should not enable an attacker to determine the session key.

Our investigation revealed not only insecure wireless networks owned by private citizens, but also company-owned wireless networks with only WEP encryption or no security at all. Many users apparently fail to recognize that radio signals from Wi-Fi devices penetrate walls, ceilings, floors, and other obstacles and that hackers can easily pick them up using standard hardware and a sniffer program.

Since numerous Web sites and readily available books detail how to crack WEP keys and extract data from Wi-Fi networks, wireless links protected by WEP alone can no longer be considered safe. Casual home users who generate little packet traffic arguably can continue using WEP for a limited time, as it can take several days to capture the one to six million packets needed to break a WEP key. Companies, however, generate considerably more traffic on wireless links and should therefore implement additional security as soon as possible.[6]

WIRELESS SECURITY OPTIONS

Several alternative security solutions to WEP are available, the most popular and useful being Wi-Fi protected access, virtual private networks, and captive portals.

Wi-Fi protected access

The Wi-Fi Alliance (www.wi-fi.org) created the interim WPA standard, which specifies security enhancements for authentication, access control, replay prevention, message integrity, message privacy, and key distribution in existing Wi-Fi systems. Applicable to home as well as enterprise users, the standard is designed to run on existing hardware as a software upgrade and is forward-compatible with the new IEEE 802.11i standard.

Features. To improve message protection, WPA utilizes the Temporal Key Integrity Protocol, which is designed to address all known attacks against, and deficiencies in, the WEP algorithm. TKIP defends against replay and weak key attacks, detects message modification, and avoids key reuse.

To improve user authentication and access control, WPA implements the Extensible Authentication Protocol (EAP) and the IEEE 802.1x standard for port-based access control. This framework uses Radius (Remote Authentication Dial-in User Service), a central authentication server, to authenticate each user on the network.

Rather than being an authentication protocol, EAP is a transport protocol tailored to the needs of upper-layer authentication protocols. It provides a plug-in architecture for numerous popular ULA protocols in use today.[3] These protocols facilitate a mutual authentication exchange between a mobile station and the Radius server residing on the network. They also generate keys for use on the wireless link between the mobile station and access point.

In a home or small office/home office (SOHO) environment, where there is no central Radius server or EAP framework, WPA runs in a special home mode, called *preshared key*, for which a user must enter a password before a mobile station can join the network. ULA is not supported in preshared key mode.

Key-scheduling flaw. WPA obtains the 128-bit *temporal key* from the EAP framework during authentication and inputs it into a key hash function together with the 48-bit *transmitter address* and a 48-bit *initialization vector*. The hash function outputs a 128-bit WEP key, or packet key. This key is used for only one WEP frame since the initialization vector is implemented as a counter that increases with each new package.

Because each package contains the initialization vector in cleartext, an attacker can obtain all utilized initialization vectors.[7] For example, let IV32 denote the most significant 32 bits of the 48-bit initialization vector. Given two WEP keys based on the same IV32, an attacker can use software to determine the temporal key. It typically takes about 30 hours to run such a program on a 2.53-GHz Intel Pentium 4, but the processing time is only six or seven minutes when four or more WEP keys based on the same IV32 are available.

WPA security relies wholly on the secrecy of all WEP (packet) keys. The attacker can determine the WEP keys based on the temporal key and decrypt all packets generated during the complete session. The attack does not imply that WPA is broken, but it underlines the importance of keeping every WEP key secret. In a well-designed system, cracking two packet keys should not enable an attacker to determine the session key. Thus, it can be said that WPA has a serious design weakness.

Interoperability problems. The Transport Layer Security protocol is the default ULA method for WPA. TLS (also denoted as EAP-TLS) is based on the Secure Socket Layer 3.0 protocol specification. SSL is a public-key, cryptography-based confidentiality mechanism.

While the Wi-Fi Alliance has recommended that all WPA products should support TLS, manufacturers can choose another ULA method. Although TLS will likely be the most popular method, using

different ULA protocols creates interoperability problems between different systems. If most enterprise WPA systems use TLS, it could become the most popular ULA protocol in systems implementing the new 802.11i security standard.

Denial-of-service attacks. The goal of a DoS attack is to deny legitimate users access to a resource by disrupting or attacking the resource itself. For example, an attacker could generate numerous connection requests to a server, effectively blocking access to this server for many hours.

DoS attacks carried out at layer 2—the media access control (MAC) layer—of Wi-Fi networks exploit a management frame's lack of encryption and integrity protection even when WPA or 802.11i is utilized. An attacker can easily forge management packets and send disassociation or deauthentication packets to the mobile station or access point, thereby denying or delaying legitimate packets. Radio-frequency-based DoS attacks at a Wi-Fi network's physical layer are also possible. There are no efficient countermeasures against DoS attacks.[3]

Virtual private networks

A *virtual private network* is a security mechanism that superimposes a private network on top of a public network, such as the Internet. Most VPNs create point-to-point connections between a user and server that serve as tunnels through the public network. Various encryption techniques ensure that only the entities at each end of the tunnels can read the transmitted messages.

VPN tunnels are often used to connect employees to their company's intranet. One end of the tunnel is a VPN software client on the employee's laptop, while the other end is the VPN server software running on the company's computer. A VPN tunnel is particularly useful to an employee connecting from a Wi-Fi hotspot whose access points and wired network are outside the company firewall. After authentication, the VPN server opens a port in the firewall to give the employee intranet access through the VPN tunnel.

While WEP and WPA encrypt data only on the wireless link, VPNs keep the data encrypted all the way from the wireless-enabled laptop to the VPN server. Hence, the hotspot owner cannot read the transmitted messages.

VPN limitations. A VPN tunnel is ideal if a laptop client wants to communicate with only one server. If the client must communicate with multiple servers, however, it is necessary to establish a VPN tunnel to each server.

Another limitation is that a user who wants to browse Web sites must often turn off the VPN because most Web servers do not support it. This problem can be solved by letting all traffic from a laptop client go through a company's VPN server. To enable Web browsing, the traffic must first go through the VPN tunnel and the company intranet, before going back out on the Internet. This solution, however, might not be very efficient.

VPNs keep data encrypted all the way from the wireless-enabled laptop to the VPN server.

Incompatible implementations. The main problem with VPNs is different, incompatible implementations. Some are based on the Layer 2 Tunneling Protocol and Internet Protocol security. L2TP extends the Point-to-Point Protocol by facilitating the tunneling of PPP packets across an intervening network. IPsec provides privacy protection, integrity checking, and replay protection as well as mutual authentication through the use of client and server certificates. There also are many VPN implementations that are based on IPsec alone (without L2TP).

Other implementations are based on Microsoft's Point-to-Point Tunneling Protocol (PPTP) and one of two authentication protocols: the Microsoft Challenge Authentication Protocol (MSCHAP2) or TLS. PPTP also utilizes Microsoft Point-to-Point Encryption based on the stream cipher RC4, but it is not considered very secure.[8] Security experts maintain that IPsec-based VPN implementations offer the best security,[9] although some are vulnerable to man-in-the-middle attacks.

Many observers claim that IPsec VPNs will prevail in the long run. Others claim that IPsec is simply too complicated to install, and that simpler solutions are needed. Currently, it is not even possible to guarantee that two different implementations of IPsec VPNs will be able to communicate. Also, users having to install their own VPN clients often have problems configuring the clients.

Captive portals

A *captive portal* is a router or a gateway host that will not allow traffic to pass before user authentication.[10]

Consider the scenario in which a user with a mobile station wants to connect to a wired network through a Wi-Fi access point and the network has a Dynamic Host Configuration Protocol server. The following steps then define a portal's operation:

- let the mobile station receive an IP address from the DHCP server via a Wi-Fi link;
- block traffic, except to the captive portal server on the wired network;

Rogue Access Points

Because small Wi-Fi networks with only a few access points are relatively easy to install, many users have Wi-Fi networks at home. Many of these wireless network owners want to have the same wireless access at work, and the more adventurous ones buy access points on their own and connect them to their corporation's intranet without permission.

Most of these *rogue access points* are consumer grade with user-friendly default configurations and security features turned off. Unlike enterprise-class access points, which include management interfaces to the wired network and broadcast themselves when installed, rogue access points may not identify themselves on the wired network—in fact, they can be totally silent and transparent to the network administrator.

A rogue access point allows just about anyone, including a wireless hacker, to access the corporate network. Even though VPNs and firewalls control access through the authorized access points, the rogue access point can be wide open with WEP or WPA disabled.

Most employees are unaware of the risks that installing rogue access points pose. They may make no or only minimal changes to the access points' default settings. Consequently, some rogue access points can even be hidden to wired-side sniffers because they duplicate the MAC address of the employee's laptop. This duplication is the result of the mandatory configuration for some consumer-grade access points when installed on a home cable or digital-subscriber-line modem.

Consumer-grade access points often contain a DHCP server that is turned on by default. Installing a rogue access point can result in two DHCP servers on the same network segment, creating havoc. This event is likely to be discovered quickly, but discovering a rogue access point is more difficult if its DHCP server is turned off.

There are no standard techniques for finding rogue access points, but commercial tools are available for this purpose. Often, the software for detecting rogue access points is part of the platform used to manage large Wi-Fi networks. All network administrators should war-walk on a regular basis to detect rogue access points.

- redirect any Web traffic from the mobile station to the captive portal;
- return a Web page displaying terms of use, billing information, or a login screen;
- once the user has accepted the terms, or logged in, allow access.

There are at least three different ways to use a captive portal. The first limits access to a set of known users defined by usernames and passwords, the second requires payment before service is established, and the third simply displays the terms of use before granting access.

Many portals only encrypt usernames and passwords during the authentication phase, and thus transmit all user data in the clear. Some portals do not even encrypt usernames and passwords. Many hotspot operators only use portals to obtain payment and leave it to users to protect their own data, sometimes without informing them.

Some portals only display the terms of use, and users often can access the Internet after simply entering their name. In this case, the name and unique MAC address of the user's mobile station—typically a laptop—serve as identifiers. Because all MAC addresses transmit in the clear, it is possible to determine another mobile station's MAC address and change it using a driver GUI in Microsoft Windows or the ifconfig command in Linux and BSD. Thus, a wireless hacker can get anonymous Internet access and shift the blame for any wrongdoing to others.

RECOMMENDATIONS

Unfortunately, no universal solution to Wi-Fi security problems is presently available. Both WPA and VPNs have potential, but their use often creates configuration and interoperability problems for users. It is possible, however, to draw some conclusions and offer a few recommendations.

WPA

We strongly urge both SOHO users and corporations to stop using WEP. SOHO users should upgrade to WPA in preshared key mode, as running it does not require any infrastructure. Corporations could upgrade to full WPA including use of a Radius server for authentication, but should only deploy if they plan to implement the new IEEE 802.11i security standard once it becomes available. It is therefore important to buy Wi-Fi equipment that can be upgraded from WPA to the 802.11i standard. Because WPA has some documented weaknesses, a corporation using WPA as an interim solution must keep up with WPA research.

Companies should avoid connecting access points using only WEP directly to their internal networks. Instead, they should connect all Wi-Fi access points in a wireless network to a separate wired network segment outside a firewall, and they should consider this network segment to be insecure. Companies should maintain this practice when updating to WPA. In the future, when an 802.11i security solution is available, it may be possible to connect the access points directly to the company's internal network.

The "Rogue Access Points" sidebar describes the serious security risk posed by users who buy their own access points and connect them to their company's intranet without permission.

VPNs

A VPN can be a good security solution for a large company, especially since its IT department can preinstall VPN clients on the employees' laptops. The VPN secures the network connections from the laptops all the way to the VPN server on the company network.

It is more difficult to implement a VPN in a university or other environment where users must install their own VPN clients. Users are likely to employ multiple operating systems and OS configurations, requiring numerous VPN clients. Even if it were possible to find clients that are stable on all platforms, many users would have trouble installing and configuring them.

Captive portals

Captive portals are very useful—many hotels, for example, use them to ensure that their customers pay for wireless Internet access. However, the lack of independent analysis and quality documentation makes it hard to assess a particular solution's level of security. Because some portals offer only authentication without any encryption of passwords or user data, it is important to verify that a portal offers the required security services as well as to obtain information about its cryptographic techniques and protocols.

Hotspots

Because Wi-Fi networks make it easy for users to connect to the Internet while on the road, hotspots continue to pop up everywhere. However, as our study revealed, many of these hotspots do not support WPA. Therefore, users who want to connect to their company should use a VPN. In fact, regardless of the security a hotspot offers, a VPN is the most secure way to communicate because it keeps the data encrypted on the wired network, denying the hotspot owner any access to the transmitted information.

SSL and SSH. Wi-Fi users can use SSL and the Secure Shell protocols in a hotspot employing a captive portal with no encryption of user data. HTTPS uses SSL to enable secure access to Web pages. Some mail protocols, such as version 3 of the Post Office Protocol and the Internet Message Access Protocol, also employ SSL.

SSH authenticates and encrypts remote command-line connections; it is thus a secure alternative to rlogin. The protocol utilizes public-key cryptography like SSL but does not rely on a trusted authority to issue certificates. An SSH tunnel between a laptop and a server on the wired Internet can be used to encrypt all types of incoming and outgoing traffic. While SSL only works from program to program, SSH can connect two arbitrary ports through a tunnel. However, only users with access to a server that runs SSH can employ an SSH tunnel.

The main problem with the SSL/SSH solution is that it requires configuration of application software and SSH clients. It may not be difficult to encrypt all e-mail and Web traffic. Advanced users might be able to configure an SSH tunnel, but this is nontrivial for the average user, at least on some platforms. Of course, a corporation distributing fully configured laptops to its employees can use SSL and SSH.

Personal firewall. All Wi-Fi users should install a personal firewall on their laptops, not only to help prevent others at nearby hotspots from accessing their devices but also as part of a broad-based defense against hackers residing on other parts of the Internet.

Researchers continue to develop more robust security solutions for Wi-Fi networks. In the meantime, because IT personnel do not control access points in home networks, a wireless hacker can steal company data or upload malicious software through local machines. Companies should carefully consider this scenario before allowing employees to access corporate data through wireless devices at home or on the road. ■

References

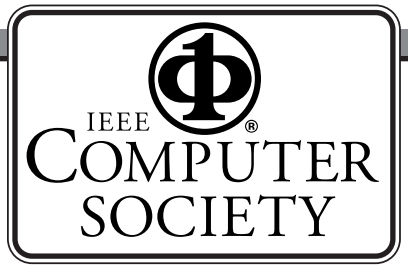
1. M.S. Gast, *802.11 Wireless Networks: The Definitive Guide,* O'Reilly, 2002.
2. S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets & Solutions*, 4th ed., McGraw-Hill/Osborne, 2003.
3. J. Edney and W.A. Arbaugh, *Real 802.11 Security: Wi-Fi Protected Access and 802.11i,* Addison-Wesley, 2004.
4. L. Barken, *How Secure Is Your Wireless Network? Safeguarding Your Wi-Fi LAN,* Prentice Hall PTR, 2004.
5. B. Schneier, *Secrets & Lies: Digital Security in a Networked World,* John Wiley & Sons, 2000.
6. A. Engst and G. Fleishman, *The Wireless Networking Starter Kit: The Practical Guide to Wi-Fi Networks for Windows and Macintosh,* 2nd ed., Peachpit Press, 2004.
7. V. Moen, H. Raddum, and K.J. Hole, "Weaknesses in the Temporal Key Hash of WPA," *ACM Sig-*

Mobile Mobile Computing and Comm. Rev., vol. 8, no. 2, 2004, pp. 76-83.
8. B. Schneier, "Analysis of Microsoft PPTP Version 2"; www.schneier.com/pptp.html.
9. N. Ferguson and B. Schneier, "A Cryptographic Evaluation of IPsec"; www.schneier.com/paper-ipsec.html.
10. B. Potter and B. Fleck, *802.11 Security*, O'Reilly, 2003.

Kjell J. Hole *is a professor in the Department of Informatics and a member of the Selmer Center at the University of Bergen, Norway. His research interests include network security and resource management in wireless networks. Hole received a PhD in computer science from the University of Bergen. He is a member of the IEEE and the IEEE Computer Society. Contact him at kjell.hole@ii.uib.no.*

Erlend Dyrnes *is a senior manager with Ernst & Young, Bergen, Norway, where he is responsible for all IT-audit and information security advisory services. His research focuses on the technical vulnerabilities of computing platforms, operating environments, and information systems. Dyrnes holds CISA and CISM certifications from the Information Systems Audit and Control Association (ISACA). Contact him at erlend.dyrnes@no.ey.com.*

Per Thorsheim *is a security coordinator with EDB Business Partner in Bergen, Norway. He holds CISA and CISM certifications from ISACA, and CISSP certification from the International Information Systems Security Certification Consortium. Contact him at per.thorsheim@edb.com.*



PURPOSE The IEEE Computer Society is the world's largest association of computing professionals, and is the leading provider of technical information in the field.

MEMBERSHIP Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEB SITE
The IEEE Computer Society's Web site, at **www.computer.org**, offers information and samples from the society's publications and conferences, as well as a broad range of information about technical committees, standards, student activities, and more.

OMBUDSMAN Members experiencing problems—magazine delivery, membership status, or unresolved complaints—may write to the ombudsman at the Publications Office or send an e-mail to help@computer.org.

CHAPTERS Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION

To obtain more information on any of the following, contact the Publications Office:

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

To check membership status or report a change of address, call the IEEE toll-free number, +1 800 678 4333. Direct all other Computer Society-related questions to the Publications Office.

PUBLICATIONS AND ACTIVITIES

Computer. The flaship publication of the IEEE Computer Society, *Computer* publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals. The society publishes 15 magazines and 14 research transactions. Refer to membership application or request information as noted at left.

Conference Proceedings, Tutorial Texts, Standards Documents. The IEEE Computer Society Conference Publishing Services publishes more than 175 titles every year.

Standards Working Groups. More than 150 groups produce IEEE standards used throughout the world.

Technical Committees. TCs provide professional interaction in over 30 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education. The society holds about 150 conferences each year and sponsors many educational activities, including computing science accreditation.

Multiprocessor Systems-on-Chips

Single processors may be sufficient for low-performance applications that are typical of early microcontrollers, but an increasing number of applications require multiprocessors to meet their performance goals.

Ahmed Jerraya
TIMA

Hannu Tenhunen
Royal University of Technology, Stockholm

Wayne Wolf
Princeton University

Multiprocessor systems-on-chips are one of the key applications of VLSI technology today. MPSoCs embody complex systems and enable large markets that leverage the large investments required for advanced VLSI fabrication lines.

Many applications, such as mobile systems, require single-chip implementations to meet the application's size and power consumption requirements. Other applications leverage large chips to reduce system cost. Systems-on-chips provide single-chip solutions in all these cases. SoCs are often customized to the application to improve their power/performance ratio or their cost.

Some of today's complex applications may inherently require programmability, as in the case of Java-enabled devices. Often, the only way to make the system work in a reasonable amount of time is to build programmable components. While single processors may be sufficient for low-performance applications that are typical of early microcontrollers, an increasing number of applications require multiprocessors to meet their performance goals.

MPSoCs, therefore, are increasingly used to build complex integrated systems. A multiprocessor system-on-chip is more than just a rack of processors shrunk down to a single chip. Both application requirements and implementation constraints push developers to build custom, heterogeneous architectures.

The applications that SoC designs target exhibit a punishing combination of constraints:

- not simply high computation rates, but real-time performance that meets deadlines;
- low power or energy consumption; and
- low cost.

Each of these constraints is difficult in itself, but the combination is extremely challenging. And, of course, while meeting these requirements, we can't break the laws of physics.

MPSoCs balance these competing constraints by adapting the system's architecture to the application's requirements. Putting computational power where it is needed meets performance constraints; removing unnecessary elements reduces both energy consumption and cost.

MPSoCs are not chip multiprocessors. Chip multiprocessors are components that take advantage of increased transistor densities to put more processors on a single chip, but they don't try to leverage application needs. MPSoCs, in contrast, are custom architectures that balance the constraints of VLSI technology with an application's needs.

Published by the IEEE Computer Society
0018-9162/05/$20.00 © 2005 IEEE

APPLICATIONS DRIVE SYSTEMS

SoCs are often enabled by standards. Complex chips usually need large markets to be economically viable. Designing a large chip typically costs $10 million to $20 million; expensive masks add to the nonrecurring costs of large chips. Large markets spread these fixed costs over more chips. Standards—multimedia, networking, communications—provide large markets with relatively stable requirements.

Standards committees often provide reference implementations. They may write these implementations in C or in a more abstract form such as Matlab. Running the reference implementation helps system designers understand the standard's nature.

Developers can use the reference implementation as a starting point. However, many reference implementations were written with flexibility, not performance or efficiency, in mind. Developers often must extensively modify a reference implementation to use as a system design.

Standards generally provide some freedom of implementation. They usually specify input and output characteristics rather than the algorithms used to generate the data. This allows designers to differentiate their system by improving its quality or lowering its power consumption.

This also means that designers might develop algorithms at the same time as the architecture. Algorithm designers need good estimates of implementation properties, such as power consumption and performance, to guide their decisions. System architects need to provide enough flexibility in the architecture design—programmability, bandwidth, and so on—to accommodate last-minute changes to algorithms.

SPECIALIZED PROCESSING ELEMENTS

Many applications can take advantage of specialized CPU instructions. A specialized instruction can speed up an operation while still providing the flexibility of a programmable processor. But if designing a custom CPU is hard, designing the compiler and other software support that goes with it is even harder.

A configurable processor is designed to be extensible in a number of ways: instruction set, word width, cache size, and so on. Given a set of refinements to the architecture, tools generate both the CPU hardware design and the compiler, debugger, and so on to go with that processor.

Hardwired accelerators are an alternative, complementary way to efficiently execute operations. Some functions are used regularly but in larger chunks than instructions. For example, the 8 × 8 discrete cosine transform is widely used in a number of image and video applications. An accelerator could perform the entire function without intervention.

Standards often provide primitive operations that can be performed relatively independently and do not require the flexibility of full software implementations. For such functions, accelerators are often the lowest power implementation available.

Irregular memory structures are often necessary in MPSoCs.

MEMORY SYSTEMS

MPSoCs often have heterogeneous memory systems. Some blocks of memory may be accessible by only one or a few processors. There can be several specialized memory blocks in a system, along with some more widely accessible memory. Heterogeneous memory systems are harder to program because the programmer must keep in mind what processors can access what memory blocks. But irregular memory structures are often necessary in MPSoCs.

One reason that designers resort to specialized memory is to support real-time performance. A block of memory that only some of the processors can access makes programming more difficult, but it makes it easier to figure out how the processors can interfere as they access the memory. Unpredictable memory access times can make it impossible for processing elements to finish tasks by their deadlines.

Having fewer processing elements with access to a block of memory reduces interference with time-critical accesses. Sharing many data and program elements in a large common memory makes analyzing memory system performance more difficult.

Although tools for memory system analysis are improving, system architects often build in predictability with custom memories. Custom memory architectures can reduce energy consumption, and smaller memory blocks consume less power. Smaller buses and interconnect structures, spanning fewer processing elements, also consume less power.

The complex applications that run on MPSoCs show wide variations in data loads during execution. The intersubsystem communication network design and usage are important factors that fix the performance and cost of the overall resulting design. Although considerable research effort has been directed to interconnects, this problem is still far from being solved in the design of specific multiprocessor architectures.

MPSoCs combine the difficulties of building complex hardware systems and complex software systems.

Existing work comes from two different communities:

- *Classical computer architecture with the system bus concept*. Much work has been done on bus architectures and arbitration strategies. This work is generally tightly coupled with the memory architecture. The key problem with the bus architecture is scaling when it incorporates a massive number of processors.
- *Networking, which generated the concept of networks on chips*. Key NoC concepts include distributing the communication structure and using multiple routes for data transfer. This allows creating flexible, programmable, and even reconfigurable networks. NoCs allow targeting MPSoC platforms to a much wider variety of products.

CHALLENGES AND OPPORTUNITIES

Many MPSoCs have already been designed, but advances in Moore's law present continuing challenges. MPSoCs combine the difficulties of building complex hardware systems and complex software systems.

Methodology is critical to MPSoC design. Designers of general-purpose processors do not rely as much on explicit methodologies, but the time pressures under which they design MPSoC designs often demand a more structured approach. Because they design MPSoCs more frequently than microprocessors, developers have enough experience to develop and refine their design methodologies.

Methodologies that work offer many advantages. They decrease the time it takes to design a system; they also make it easier to predict how long the design will take and how many resources it will require. Methodologies also codify techniques for improving performance and power consumption that developers can apply to many different designs.

MPSoC methodologies will necessarily be a moving target for the next decade, and they must be constantly tweaked: New technologies change the underlying components; new tools support new approaches to design.

MPSoC hardware architectures present challenges in all aspects of the multiprocessor: processing elements, memory, and interconnects.

Configurable processors with customized instruction sets are one way to improve the characteristics of processing elements; hardware/software codesign of accelerators is another technique.

Researchers have been studying memory systems for symmetric multiprocessors for many years, and MPSoCs often use heterogeneous memory systems to improve real-time performance and power consumption. However, designers often make memory system design choices based on insufficient data and inadequate design space exploration.

As designs move from buses to more general networks, developers must navigate a much larger design space. NoCs provide one promising way to modularize the interconnect design in very large chips.

A critical question in MPSoC architectures is the balance between programmability and efficiency. It's more difficult to program multiprocessors than uniprocessors, and it's more difficult to program heterogeneous multiprocessors than homogeneous multiprocessors.

MPSoCs are heterogeneous in just about every way possible: multiple instruction sets, hardwired function units, heterogeneous memory systems and address spaces, interconnects of varying bandwidth and with less than fully connected topology.

The choice of programmability features in an MPSoC is an art—and a not very well understood one at that. A more principled approach to the choice of programmable structures would be a welcome help to many MPSoC designers.

Modern ASIC design relies heavily on synthesis, both at the logic and physical levels. MPSoC design today uses more simulation and hand analysis than ASIC designs.

We do not have tools that will automatically turn a large application into a complete design. Even within a fairly narrow application, developing these complex tools is difficult, and the market for such tools is too limited to make them economically feasible. Furthermore, MPSoC-enabled applications increasingly run the gamut from cryptography to video, making it difficult to narrow the domain in which a tool must work.

Because designers have fewer synthesis tools, they need excellent simulation tools, but not enough good MPSoC simulators exist. Most multiprocessor simulators developed for general-purpose computer design can simulate only heterogeneous multiprocessors. It is often extremely difficult to modify such simulators to handle heterogeneous processing elements, memory, and interconnects.

MPSoC designers need simulators that can help them characterize their systems at multiple levels of abstraction: functional simulators for software analysis and debugging; cycle-accurate simulators

for detailed performance analysis; and power simulators for energy analysis.

Methodologies must also take into account the system specification's source. Standards committees often provide reference implementations of their standards. The good news is that these are executable specifications that researchers can run and analyze. The bad news is that they are often programmed in a style that is not well suited to SoC implementations.

Reference implementations often use dynamic memory management (such as malloc in Unix) in ways that simplify the life of the reference implementation's programmer but cause performance problems in SoCs. They also often are not optimized for performance or for power consumption. Thus, the SoC design team often must spend a considerable amount of time optimizing the code even before they target it for their particular MPSoC.

THE BREAKTHROUGH: HW/SW INTERFACE CODESIGN

The key issue when integrating an MPSoC's parts is creating a continuum between embedded software and hardware. This requires new technologies allowing HW/SW interface codesign to integrate embedded SW and HW components. The discipline of HW/SW interface codesign requires using specific programming models that can hide sophisticated HW/SW interfaces.

Most conventional parallel programming models that are used on general-purpose designs—such as the message passing interface, OpenMP, BSP, LogP, and so on—primarily target single processor subsystems and homogeneous MPSoCs like SIMD. MPSoCs require more complex HW/SW interface descriptions to describe the interactions between heterogeneous subsystems. MPSoC application programming interfaces need to specify the application-specific design constraints such as quality of service (QoS) in terms of energy consumption, runtime, cost, and reliability. The challenge will be abstracting complex heterogeneous multiprocessor platforms.

Mastering HW/SW interfaces codesign opens new vistas that will bring fundamental improvements to the design process:

- concurrent design of both hardware and embedded software, leading to a shorter time-to-market;
- modular design of hardware and software components, leading to clearer design choices when building complex systems; and
- easier global validation of embedded systems including hardware and embedded software, leading to increased reliability and improved quality of service.

Verification of the HW/SW interface itself imposes new challenges.

Additionally, HW/SW interface codesign makes several technical challenges more tractable:

HW/SW interface codesign ensures embedded software quality. Embedded software relies on the hardware platform to support complex QoS requirements and high reliability. Current practice uses an existing OS or middleware to validate nonfunctional properties of application software. These validation methods generally support real-time and delay requirements. Unfortunately, they do not support other nonfunctional properties such as intersubsystem communication bandwidth, jitter, and reliable communication.

Currently, developers validate embedded software QoS requirements only when the physical hardware prototype becomes available. This does not allow monitoring to guarantee the required QoS in a systematic manner. In such a scenario, it is even difficult to guarantee the reliability of HW/SW interface design itself. Overcoming this challenge requires a QoS-aware HW/SW interface abstraction.

Abstract HW/SW interfaces allow verification early in the design process. Verification of the HW/SW interface itself imposes new challenges. It is not sufficient to verify the interface independent of its context—the interface must be verified relative to a given hardware platform. Of course, researchers can't wait until the hardware prototype is available to carry out this verification. Researchers can use an abstract HW/SW interface to verify the correctness of the interface abstract design without using the physical prototype.

Abstract HW/SW interfaces allow using standards. A fairly general HW/SW interface model can be useful and applicable in a variety of applications. The advantage of such a general model is that developers can reuse application software, hardware components, and platform and middleware modules across different products, product families, and even application domains.

The drawback that accompanies generality is inefficiency. For applications that require only a small subset of the complete HW/SW interface functionality, a generic model carries a tremendous overhead that cost-sensitive applications can't tolerate. Using an abstract HW/SW interface architecture that is highly configurable and parameter-

ized mitigates this problem. Using an interface allows optimizing and streamlining an instance of the interface to the particular needs of a given application. This is a central advantage of the HW/SW interface concept because without an efficient method to configure and optimize the HW/SW interface, the embedded system design cannot be mastered.

Abstract HW/SW interfaces facilitate interoperability. Abstract HW/SW interfaces enable dialogues between separate teams that can belong to different companies or even different market sectors. For example, a car maker could use a standard HW/SW interface API to develop the vehicle's software while delaying selection of the hardware platform as long as possible.

IN THIS ISSUE

This issue contains four interesting articles on MPSoC design.

In "Parallelism and the ARM Instruction Set Architecture," John Goodacre and Andrew N. Sloss trace the development of the ARM architecture. Their article shows how the original RISC architecture was tuned over several generations to provide features important for embedded applications. It also describes how developers will use new multiprocessors in future generations of high-performance, low-power systems.

"Configurable Processors: A New Era in Chip Design," by Steve Leibson and James Kim, looks at the configurable processor approach to SoC design, which leverages the benefits of nanometer silicon lithography with relatively little manual effort. SoC designers can customize configurable processors to connect multiprocessors, providing even more computing power.

In "An Open Platform for Developing Multiprocessor SoCs," Mario Diaz Nava and coauthors describe a low-cost modular approach that uses emulation as an alternative to software simulation for the design and verification of complex MPSoC designs.

"Evaluating Digital Entertainment System Performance," by Marcus Levy, describes the process of benchmark design for systems-on-chips. Because of the complex nature of modern applications, benchmark design is a critical problem for both MPSoC designers and customers.

The Embedded Systems column in this issue also contributes to this look at MPSoCs. "Absolutely Positively on Time: What Would It Take?" by Edward A. Lee of UC Berkeley challenges us to develop design methodologies that will ensure that our systems operate on time.

The industrial perspective of the articles in this issue shows the rapid advances that are being made in this field for deployment in products. This issue focuses more on hardware than on software. We could fill an entire issue just on software challenges in SoCs. ■

Ahmed Jerraya *is research director at TIMA Laboratory, Grenoble, France. His research interests focus on multiprocessor system-on-chip specification, validation, and design. Jerraya received a PhD in computer sciences from the University of Grenoble. He is a member of the IEEE, SIGDA, and EDAA. Contact him at ahmed.jerraya@imag.fr.*

Hannu Tenhunen *is a professor of electronic system design at the Royal University of Technology, Stockholm, Sweden. His research interests include interconnect-centric design, mixed-signal system integration, and nanoelectronic systems. He received a PhD in electrical engineering from Cornell University. He also received an honorary doctor degree (Dr.h.c.) from Tallinn Technical University, Estonia, and holds honorary professor appointments from Fudan University and Bejing Jiatong University, China. Contact him at hannu@imit.kth.se.*

Wayne Wolf *is a professor in the Department of Electrical Engineering at Princeton University. His research interests include embedded computing, multimedia systems, VLSI, and computer-aided design. Wolf received a PhD in electrical engineering from Stanford University. He is a Fellow of the IEEE and the ACM. Contact him at wolf@princeton.edu.*

Don't Put It Off Any Longer

STRENGTHEN YOUR PROFESSIONAL QUALIFICATIONS TODAY

New Testing Centers Now Open

The IEEE Computer Society's Certified Software Development Professional Program recently added 51 new testing centers in 32 new countries in Europe, Asia, and Latin America, in addition to testing centers throughout the US and Canada. With so many testing centers, it's more convenient than ever for software engineers to take the exam and earn the CSDP credential.

CSDP TESTING CENTERS

ASIA

CHINA
Beijing
Shanghai

INDIA
Ahmedabad
Allahabad
Bangalore
Calcutta
Chennai
Hyderabad
Mumbai
Delhi

JAPAN
Tokyo

EUROPE

ARMENIA
Yerevan

BULGARIA
Sofia

CROATIA
Zagreb

FINLAND
Helsinki

FRANCE
Paris
Toulouse

GEORGIA
Tbilisi

GERMANY
Berlin
Frankfurt
Hamburg
Munich

GREECE
Athens
Thessaloniki

HUNGARY
Budapest

ISRAEL
Tel Aviv

ITALY
Milan

IRELAND
Dublin

KAZAKHSTAN
Alma-Ata

LITHUANIA
Vilnius

NETHERLANDS
Arnhem

PORTUGAL
Lisbon

ROMANIA
Bucharest

RUSSIA
Moscow
St. Petersburg

SPAIN
Barcelona
Madrid

SWITZERLAND
Geneva

TURKEY
Ankara
Istanbul
Izmir

UKRAINE
Kiev

UNITED KINGDOM
London
Twickenham

UZBEKISTAN
Tashkent

LATIN AMERICA

ARGENTINA
Buenos Aires

BOLIVIA
La Paz

BRAZIL
Belo Horizonte
Brasilia
Curitiba
Puerto Alegre
Recife
Rio de Janeiro
Sao Paulo

CHILE
Santiago

COLOMBIA
Bogota
Cali

DOMINICAN REPUBLIC
Santo Domingo

GUATEMALA
Guatemala City

MEXICO
Guadalajara
Mexico City
Monterrey

PANAMA
Panama City

PERU
Lima

VENUZUELA
Caracas

For more information, visit www.computer.org/certification or email certification@computer.org.

Fall Testing Window: 1 September - 30 November 2005
Application Deadline: 1 September.

IEEE

IEEE COMPUTER SOCIETY

IEEE Computer Society Certified Software Development Professional

csdp

Parallelism and the ARM Instruction Set Architecture

Leveraging parallelism on several levels, ARM's new chip designs could change how people access technology. With sales growing rapidly and more than 1.5 billion ARM processors already sold each year, software writers now have a huge range of markets in which their ARM code can be used.

John Goodacre
Andrew N. Sloss
ARM

Over the past 15 years, the ARM reduced-instruction-set computing (RISC) processor has evolved to offer a family of chips that range up to a full-blown multiprocessor. Embedded applications' demand for increasing levels of performance and the added efficiency of key new technologies have driven the ARM architecture's evolution.

Throughout this evolutionary path, the ARM team has used a full range of techniques known to computer architecture for exploiting parallelism. The performance and efficiency methods that ARM uses include variable execution time, subword parallelism, digital signal processor-like operations, thread-level parallelism and exception handling, and multiprocessing.

The developmental history of the ARM architecture shows how processors have used different types of parallelism over time. This development has culminated in the new ARM11 MPCore multiprocessor.

RISC FOR EMBEDDED APPLICATIONS

Early RISC designs such as MIPS focused purely on high performance. Architects achieved this with a relatively large register set, a reduced number of instruction classes, a load-store architecture, and a simple pipeline. All these now fairly common concepts can be found in many of today's modern processors.

The ARM version of RISC differed in many ways, partly because the ARM processor became an embedded processor designed to be located within a system-on-chip device.[1] Although this kept the main design goal focused on performance, developers still gave priority to high code density, low power, and small die size.

To achieve this design, the ARM team changed the RISC rules to include variable-cycle execution for certain instructions, an inline barrel shifter to preprocess one of the input registers, conditional execution, a compressed 16-bit Thumb instruction set, and some enhanced DSP instructions.

- *Variable cycle execution*. Because it is a load-store architecture, the ARM processor must first load data into one of the general-purpose registers before processing it. Given the single-cycle constraint the original RISC design imposed, loading and storing each register individually would be inefficient. Thus, the ARM ISA instructions specifically load and store multiple registers. These instructions take variable cycles to execute, depending on the number of registers the processor is transferring. This is particularly useful for saving and restoring context for a procedure's prologue and epilogue. This directly improves code density, reduces instruction fetches, and reduces overall power consumption.
- *Inline barrel shifter*. To make each data processing instruction more flexible, either a shift or rotation can preprocess one of the source registers. This gives each data processing instruction more flexibility.

Published by the IEEE Computer Society

- *Conditional execution*. An ARM instruction executes only when it satisfies a particular condition. The condition is placed at the end of the instruction mnemonic and, by default, is set to always execute. This, for example, generates a savings of 12 bytes—42 percent—for the greatest common divisor algorithm implemented with and without conditional execution.
- *16-bit Thumb instruction set*. The condensed 16-bit version of the ARM instruction set allows higher code density at a slight performance cost. Because the Thumb 16-bit ISA is designed as a compiler target, it does not include the orthogonal register access of the ARM 32-bit ISA. Using the Thumb ISA can achieve a significant reduction in program size.

  In 2003, ARM announced its Thumb-2 technology, which offers a further extension to code density. This technology increases the code density by mixing both 32- and 16-bit instructions in the same instruction stream. To achieve this, the developers incorporated unaligned address accesses into the processor design.
- *Enhanced DSP instructions*. Adding these instructions to the standard ISA supports flexible and fast 16 × 16 multiply and arithmetic saturation, which lets DSP-specific routines migrate to ARM. A single ARM processor could execute applications such as voice-over-IP without the requirement of having a separate DSP. The processor can use one example of these instructions, SMLAxy, to multiply the top or bottom 16 bits of a 32-bit register. The processor could multiply the top 16 bits of register r1 by the bottom 16 bits of register r2 and add the result to register r3.

Nonsaturated (ISA v4T)	Saturated (ISA v5TE)
PRECONDITION	**PRECONDITION**
r0=0x00000000	r0=0x00000000
r1=0x70000000	r1=0x70000000
r2=0x7fffffff	r2=0x7fffffff
ADDS r0,r1,r2	QADD r0,r1,r2
POSTCONDITION	**POSTCONDITION**
result is **negative**	result is **positive**
r0=**0xefffffff**	r0=**0x7fffffff**

Figure 1. Nonsaturated and saturated addition. Saturation is particularly useful for digital signal processing because nonsaturations would wrap around when the integer value overflowed, giving a negative result.

Figure 1 shows how saturation can affect the result of an ADD instruction.[2] Saturation is par-

A Short History of ARM

Formed in 1990 as a joint venture with Acorn Computers, Apple Computers, and VLSI Technology (which later became Philips Semiconductor), ARM started with only 12 employees and adopted a unique licensing business model for its processor designs. By licensing rather than manufacturing and selling its chip technology, ARM established a new business model that has redefined the way industry designs, produces, and sells microprocessors. Figure A shows how the ARM product family has evolved.

The first ARM-powered products were the Acorn Archimedes desktop computer and the Apple Newton PDA. The ARM processor was designed originally as a 32-bit replacement for the MOS Technologies 6502 processor that Acorn Computers used in a range of desktops designed for the British Broadcasting Corporation. When Acorn set out to develop this new replacement processor, the academic community had already begun considering the RISC architecture. Acorn decided to adopt RISC for the ARM processor. ARM's developers originally tailored the ARM instruction set architecture to efficiently execute Acorn's BASIC interpreter, which was at the time very popular in the European education market.

The ARM1, ARM2, and ARM3 processors were developed by Acorn Computers. In 1985, Acorn received production quantities of the first ARM1 processor, which it incorporated into a product called the ARM Second Processor, which attached to the BBC Microcomputer through a parallel communication port called "the tube." These devices were sold mainly as a research tool. ARM1 lacked the common multiply and divide instructions, which users had to synthesize using a combination of data processing instructions. The condition flags and the program counter were combined, limiting the effective addressing range to 26 bits. ARM ISA version 4 removed this limitation.

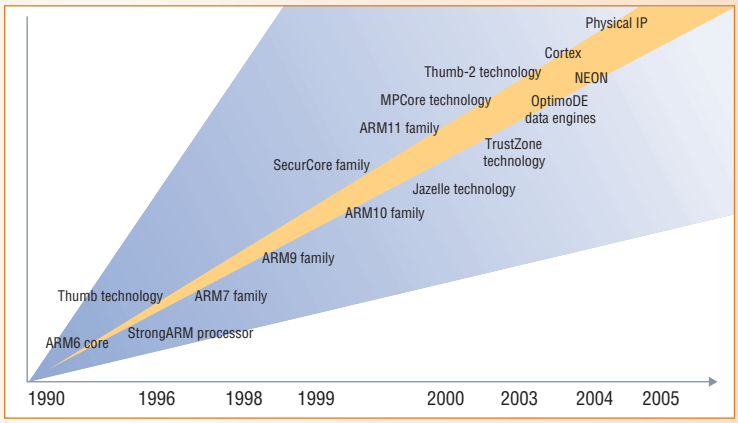


Figure A. Technology evolution of the ARM product family.

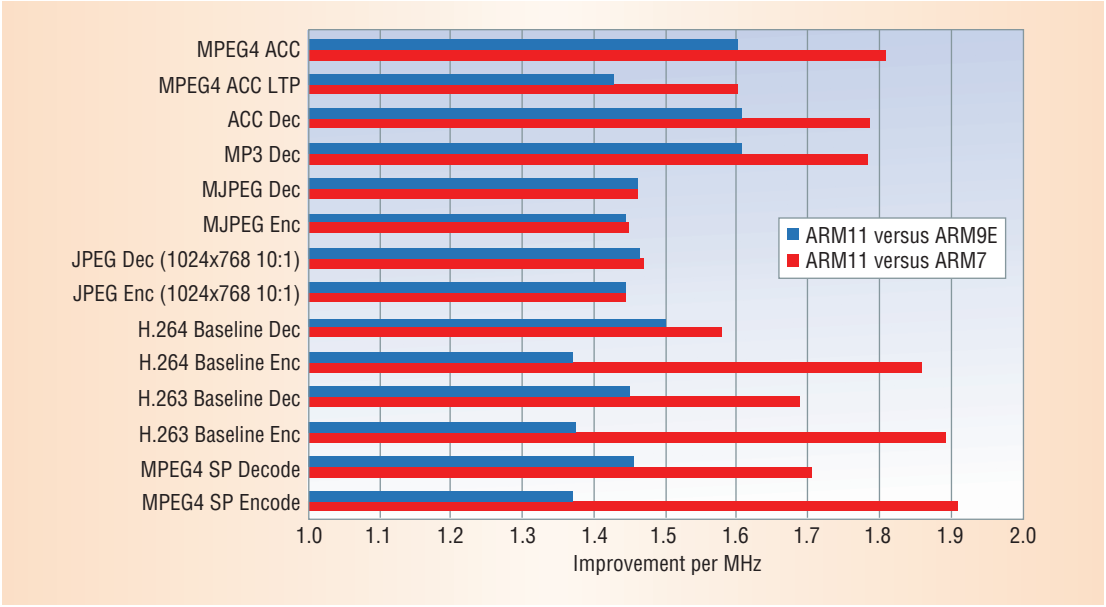


Figure 2. SIMD versus non-SIMD power consumption. The lightweight ARM implementation of SIMD reduces gate count, hence significantly reducing die size, power, and complexity.

ticularly useful for digital signal processing because nonsaturations would wrap around when the integer value overflowed, giving a negative result. A saturated QADD instruction returns a maximum value without wrapping around.

DATA-LEVEL PARALLELISM

Following the success of the enhanced DSP instructions introduced in the v5TE ISA, ARM introduced the ARMv6 ISA in 2001. In addition to improving both data- and thread-level parallelism, other goals for this design included enhanced mathematical operations, exception handling, and endian-ness handling.

An important factor influencing the ARMv6 ISA design involved increasing DSP-like functionality for overall video handling and 2D and 3D graphics. The design had to achieve this improved functionality while still maintaining very low power consumption. ARM identified the single-instruction, multiple-data architecture as the means for accomplishing this.

SIMD is a popular technique for providing data-level parallelism without compromising code density and power. A SIMD implementation requires relatively few instructions to perform complex calculations with minimum memory accesses.

Due to a careful balancing of computational efficiency and low power, ARM's SIMD implementation involved splitting the standard 32-bit data path into four 8-bit or two 16-bit slices. This differs from many other implementations, which require additional specialized data paths for SIMD operations.

Figure 2 shows the improvements in MHz that various codecs require when using the ARMv6 SIMD instructions introduced in the ARM11 processor.

The lightweight ARM implementation of SIMD reduces gate count, hence significantly reducing die size, power, and complexity. Further, all the SIMD instructions execute conditionally.

To improve handling of video compression systems such as MPEG and H.263, ARM also introduced the sum of absolute differences (SAD) concept, another form of DLP instructions. Motion estimation compares two blocks of pixels, R(I,j) and C(I,j), by computing

$$SAD = \sum | R(I,j) - C(I,j) |$$

Smaller SAD values imply more similar blocks. Because motion estimation performs many SAD tests with different relative positions of the R and C blocks, video compression systems require very fast and energy-efficient implementations of the sum-of-absolute-differences operation.

The instructions USAD8 and USADA8 can compute the absolute difference between 8-bit values. This is particularly useful for motion-video-compression and motion-estimation algorithms.

THREAD-LEVEL PARALLELISM

We can view threads as processes, each with its own program counter and register set, while having the advantage of sharing a common memory space. For thread-level parallelism, ARM needed to improve exception handling to prepare for the increased complexity in handling multithreading on multiple processors. These requirements added inherent complexity in the interrupt handler, scheduler, and context switch.

One optimization extended the exception handling instructions to save precious cycles during the time-critical context switch. ARM achieved this by adding three new instructions to the instruction set, as Table 1 shows.

Programmers can use the change processor state (CPS) instruction to alter processor state by setting

the current program status register to supervisor mode and disabling fast interrupt requests, as the code in Figure 3 shows. Whereas the ARMv4T ISA required four instructions to accomplish this task, the ARMv6 ISA requires only two.

Programmers can use the save return state (SRS) instruction to modify the *saved program status register* in a specific mode. The updating of SPSR in a particular ARMv4T ISA mode involved many more instructions than in the ARMv6 ISA. This new instruction is useful for handling context switches or preparing to return from an exception handler.

MULTIPROCESSOR ATOMIC INSTRUCTIONS

Earlier ARM architectures implemented semaphores with the swap instruction, which held the external bus until completion. Obviously, this was unacceptable for thread-level parallelism because one processor could hold the entire bus until completion, disallowing all other processors. ARMv6 introduced two new instructions—load-exclusive LDREX and store-exclusive STREX—which take advantage of an exclusive monitor in memory:

- LDREX loads a value from memory and sets the exclusive monitor to watch that location, and
- STREX checks the exclusive monitor and, if no other write has taken place to that location, performs the store to memory and returns a value to indicate if the data was written.

Thus, the architecture can implement semaphores that do not lock the system bus that grants other processors or threads access to the memory system.

The ARM11 microarchitecture was the first hardware implementation of the ARMv6 ISA. It has an eight-stage pipeline with separate parallel pipelines for the load/store and multiply/accumulate operations. With the parallel load/store unit the ARM1136J-S processor can continue executing without waiting for slower memory—a main gating factor for processor performance.

In addition, the ARM1136J-S processor has physically tagged caches to help with thread-level parallelism—as opposed to the virtually tagged caches of previous ARM processors—which considerably benefits context switches, especially when running large operating systems.

A virtually tagged cache must be flushed every time a context switch takes place because the cache contains old virtual-to-physical translations. In the ARM11, the *memory management unit* logic resides between the level 1 cache and the processor core. The reduction in cache flushing has the additional benefit of decreasing overall power consumption by reducing the external memory accesses that occur in a virtually tagged cache. The physically tagged cache increases overall performance by about 20 percent.

Table 1. Exception handling instructions in the ARMv6 architecture.

Instruction	Description	Action
CPS	Change processor state	CPS<effect> <iflags>,{,#mode} CPS #<mode> CPSID <flags> CPSIE <flags>
RFE	Return from exception	RFE<addressing_mode> Rn!
SRS	Save return state	SRS<addressing_mode>,#<mode>{!}

```
ARMv4T ISA                          ARMv6 ISA
; Copy CPSR                        ; Change processor state and modify
    MRS   r3, CPSR                  ; select bits
; Mask mode and FIQ interrupt           CPSIE   f, #SVC
    IC    r3, r3, #MASK|FIQ
; Set Abort mode and enable FIQ
    ORR   r3, r3, #SVC|nFIQ
; Update the CPSR
    MSR   CPSR_c, r3
```

Figure 3. ARMv6 ISA change processor state instruction compared with the ARMv4T architecture.

INSTRUCTION-LEVEL PARALLELISM

In ILP, the processor can execute multiple instructions from a single sequence of instructions concurrently. This form of parallelism has significant value in that it provides additional overall performance without affecting the software programming model.

Obviously, ILP puts more emphasis on the compiler that extracts it from the source code and schedules the instructions across the superscalar core. Although potentially simplifying otherwise overly complex hardware, an excessive drive to extract ILP and achieve high performance through increased MHz has increased hardware complexity and cost.

ARM has remained the processor at the edge of the network for several years. This area has always seen the most rapid advancements in technology, with continuous migration away from larger computer systems and toward smaller ones. For example, technology developed for mainframes a decade or two ago is found in desktop computers today. Likewise, technologies developed for the desktop five years ago have begun appearing in consumer and network products. For example, symmetric multiprocessing (SMP) is appearing today in both desktop and embedded computers.

Continued demand for performance at low power has led to minimizing the overall power budget by adding multiple processors and accelerators.

PERFORMANCE VERSUS POWER REQUIREMENTS

For several years, the embedded processor market inherited technology matured in desktop computing as consumers demanded similar functionality in their embedded devices. The continued demand for performance at low power has, however, driven slightly different requirements and led to the overall power budget being minimized by adding multiple processors and accelerators within an embedded design. Today, the demand for high levels of general-purpose computing drives using SMP as the application processor in both embedded and desktop systems.

In 2004, both the embedded and desktop markets hit the cost-performance-through-MHz wall. In response, developers began embracing potential solutions that require SMP processing to avoid the following pitfalls:

- *High MHz costs energy.* Increasing a processor's clock rate has a quadratic effect on power consumption. Not only does doubling the MHz double the dynamic power required to switch the logic, it also requires a higher operating voltage, which increases at the square of the frequency. Higher frequencies also add to design complexity, greatly increasing the amount of logic the processor requires.
- *Extracting ILP is complex and costly.* Using hardware to extract ILP significantly raises the cost in silicon area and design complexity, further increasing power consumption.
- *Programming multiple independent processors is nonportable and inefficient.* As developers use more processors, often with different architectures, the software complexity escalates, eliminating any portability between designs.

In mid-2004, PC manufacturers and chip makers made several announcements heralding the end of the MHz race in desktop processors and championing the use of multicore SMP processors in the server realm, primarily through the introduction of hyperthreading in the Intel Pentium processor. At that time, ARM announced its ARM11 MPCore multiprocessor core as a key solution to help address the demand for performance scalability.

Introduced alongside the ARM11 MPCore, a set of enhancements to the ARMv6 architecture provides further support for advanced SMP operating systems. In its move to support richer SMP-capable operating systems, ARM applied these enhancements, known as ARMv6K or AOS (for Advanced OS Support), across all ARMv6-architecture-based application processors to provide a firm foundation for embedded software.

The ARM11 multiprocessor also addressed the SMP system design's two main bottlenecks:

- interprocessor communication with the integration of the new ARM Generic Interrupt Controller (GIC), and
- cache coherence with the integration of the Snoop Control Unit (SCU), an intelligent memory-communication system.

These logic blocks deliver an efficient, hardware-coherent single-core SMP processor that manufacturers can build cost-effectively.

PREPARATIONS FOR ARM MULTIPROCESSING

To fully realize the advantages of a multiprocessor hardware platform in general-purpose computing, ARM needed to provide a cache-coherent, symmetric software platform with a rich instruction set. ARM found that a few key enhancements to the current ARMv6 architecture could offer the significant performance boost it sought.

Enhanced atomic instructions

Researchers can use the ARMv6 load-and-store exclusives to implement both swap-based and compare-and-exchange-based semaphores to control access to critical data. In the traditional server computing world of SMP there has, however, been significant software investment in optimizing SMP code using *lock-free synchronization*. This work has been dominated by the x86 architecture and its atomic instructions that developers can use to compare and exchange data.

Many favored using the Intel cmpxchg8b instruction in these lock-free routines because it can exchange and compare 8 bytes of data atomically. Typically, this involved 4 bytes for payload and 4 bytes to distinguish between payload versions that could otherwise have the same value—the so-called A-B-A problem.

The ARM exclusives provide atomicity using the data address rather than the data value, so that the routines can atomically exchange data without experiencing the A-B-A problem. Exploiting this would, however, require rewriting much of the existing two-word exclusive code. Consequently, ARM added instructions for performing load-and-

store exclusives using various payload sizes—including 8 bytes—thus ensuring the direct portability of existing multithreaded code.

Improved access to localized data

When an OS encounters the increasing number of threaded applications typical in SMP platforms, it must consider the performance overheads of associating thread-specific state with the currently executing thread. This can involve, for example, knowing which CPU a thread is executing on, accessing kernel structures specific to a thread, and enabling thread access to local storage. The AOS enhancements add registers that help with these SMP performance aspects.

CPU number. Using the standard ARM system coprocessor interface, software on a processor can execute a simple, nonmemory-accessing instruction to identify the processor on which it executes. Developers use this as an index into kernel structures.

Context registers. SMP operating systems handle two key demands from the kernel when providing access to thread-specific data. The ARMv6K architecture extensions define three additional system coprocessor registers that the OS can manage for whatever purpose it sees fit. Each register has a different access level:

- user and privileged read/write accessible;
- read-only in user, read/write privileged accessible; and
- privileged only read/write accessible.

The exact use of these registers is OS-specific. In the Linux kernel and GNU toolchain, the ARM application binary interface has assigned these registers to enable *thread local storage*. A thread can use TLS to rapidly access thread-specific memory without losing any of the general-purpose registers.

To support TLS in C and C++, the new keyword *thread* has been defined for use in defining and declaring a variable. Although not an official extension of the language, using the keyword has gained support from many compiler writers. Variables defined and declared this way would automatically be allocated locally to each thread:

```
__thread int i;
__thread struct state s;
extern __thread char *p;
```

Supporting TLS is a key requirement for the new Native Posix Thread Library (NPTL) released as part of the Linux 2.6 kernel. This Posix thread library provides significant performance improvements over the old Linux pthread library.

Power-conscious spin-locks

Another SMP system cost involves the synchronization overhead required when processors must access shared data. At the lowest abstraction level in most SMP synchronization mechanisms, a *spin-lock* software technique uses a value in memory as a lock. If the memory location contains some predefined value, the OS considers the shared resource locked, otherwise it considers the resource unlocked. Before any software can access the shared resource, it must acquire the lock—and an atomic operation must acquire it. When the software finishes accessing the resource, it must release the lock.

Spin-lock causes the processor to spin around a tight loop while continually attempting to acquire a lock.

In an SMP OS, processors often must wait while another processor holds a lock. The spin-lock received its name because it accomplishes this waiting by causing the processor to spin around a tight loop while continually attempting to acquire the lock. A later refinement to reduce bus contention added a back-off loop during which the processor does not attempt to access the lock. In either case, in a power-conscious embedded system, these unproductive cycles obviously waste energy.

The AOS extensions include a new instruction pair that lets a processor sleep while waiting for a lock to be freed and that, as a result, consumes less energy. The ARM11 multiprocessor implements these instructions in a way that provides next-cycle notification to the waiting processor when the lock is freed, without requiring a back-off loop. This results in both energy savings and a more efficient spin-lock implementation mechanism.

Figure 4 shows a sample implementation of the spin lock and unlock code used in the ARM Linux 2.6 kernel.

Weakly ordered memory consistency

The ARMv6 architecture defined various memory consistency models for the different definable memory regions. In the ARM11 multiprocessor, spin-lock code uses coherently cached memory to store the lock value. As a multiprocessor, the ARM11 MPCore is the first ARM processor to fully expose weakly ordered memory to the programmer. The multiprocessor uses three instructions to control weakly ordered memory's side effects:

```
static inline void _raw_spin_lock(spinlock_t *lock)
{
    unsigned long tmp;

    _asm__ __volatile__(
     1:  ldrex    %0, [%1]           ; exclusive read lock
         teq      %0, #0             ; check if free
         wfene                       ; if not, wait (saves power)
         strexeq  %0, %2, [%1]       ; attempt to store to the lock
         teqeq    %0, #0             ; Were we successful ?
         bne      1b                 ; no, try again
       :   "=&r" (tmp)
       :   "r" (&lock->lock), "r" (1), "r" (0)
       :   "cc", "memory"
    );

    rmb();    // Read memory barrier stops speculative reading of payload
}             // This is NOP on MPCore since dependent reads are sync'ed

static inline void _raw_spin_unlock(spinlock_t *lock)
{
    wmb();    // data write memory barrier, ensure payload write visible
              // Ensures data ordering, but does not necessarily wait
    _asm__ __volatile__(
        str %1, [%0]                 ; Release spinlock
        mcr p15, 0, %1, c7, c10, 4   ; DrainStoreBuffer (DSB)
        sev                          ; Signal to any CPU waiting
      :  "r" (&lock->lock), "r" (0)
      :  "cc", "memory");
}
```

Figure 4. Power-conscious spin-lock. This sample implementation shows the spin-lock and unlock code used in the ARM Linux 2.6 kernel.

- *wmb()*. This Linux macro creates a write-memory barrier that the multiprocessor can use to place a marker in the sequencing of any writes around this barrier instruction. The spin-lock, for example, executes this instruction prior to unlocking to ensure that any writes to the payload data complete before the write to release the spin-lock, and hence before any other processor can acquire the lock. To ensure higher performance, the barrier does not necessarily stall the processor by flushing data. Rather it informs the load-store unit and lets execution continue in most situations.
- *rmb()*. Again from the Linux kernel, this macro places a read-memory barrier that prevents speculative reads of the payload from occurring before the read has acquired the lock. Although legal in the ARMv6 architecture, this level of weakly ordered memory can make it difficult to ensure software correctness. Thus, the ARM11 multiprocessor implements only nonspeculative read-ahead. When the possibility exists that a read will not be required, as in the spin-lock case—where there is a branch instruction between the teqeq instruction and any payload read—the read-ahead does not take place. So, for the ARM11 MPCore multiprocessor, this macro can be defined as empty.
- *DSB (Drain Store Buffer)*. The ARM architecture includes a buffer visible only to the processor. When running uniprocessor software, the processor allows subsequent reads to scan the data from this buffer. However, in a multiprocessor, this buffer becomes invisible to reads from other processors. The DSB drains this buffer into the L1 cache. In the ARM11 multiprocessor, which has a coherent L1 cache between the processors, the flush only needs to proceed as far as the L1 memory system before another processor can read the data. In the spin-lock unlock code, the processors issue the DSB immediately prior to the SEV *(Set Event)* instruction so that any processor can read the correct value for the lock upon awakening.

ARM11 MPCORE MULTIPROCESSOR

The ISA's suitability is not the only factor affecting the multiprocessor's ability to actually deliver the scalability promises of SMP. If they are poorly implemented, two aspects of an SMP design can significantly limit peak performance and increase the energy costs associated with providing SMP services:

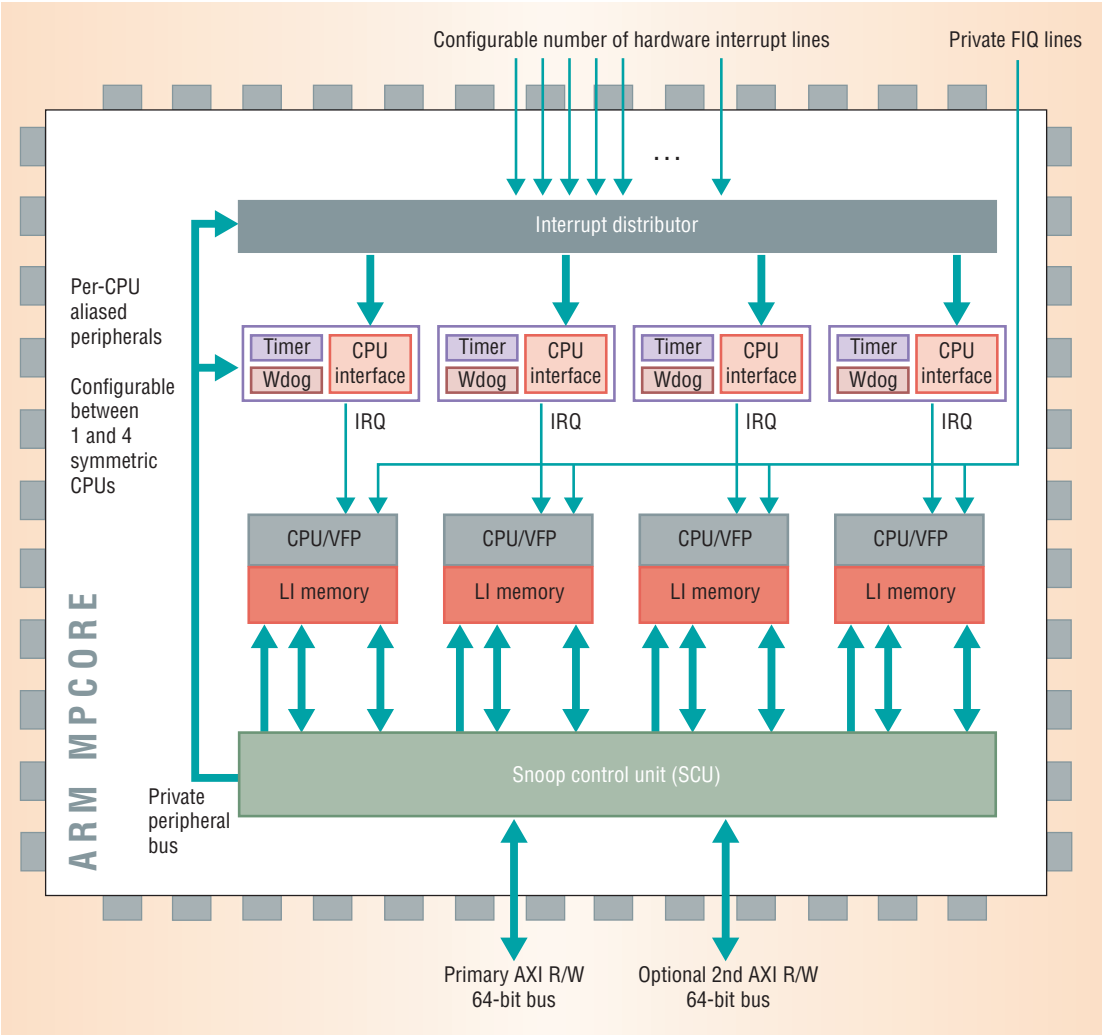


Figure 5. ARM11 MPCore. This multiprocessor integrates the new ARM GIC inside the core to make the interrupt system's access and effects much closer and more efficient.

- *Cache coherence*. Developers typically provide the single-image SMP OS with coherent caches so it can maintain performance by placing its data in cached memory. In the ARM11 multiprocessor, each CPU has its own instruction and L1 data cache. Existing coherency schemes often extend the system bus with additional signals to control and inspect other CPUs' caches. In an embedded system, the system bus often clocks slower than the CPU. Thus, besides placing a bottleneck between the processor and its cache, this scheme significantly increases the traffic and hence the energy the bus consumes. The ARM11 MPCore addresses these problems by implementing an intelligent SCU between each processor. Operating at CPU frequency, this configuration also provides a very rapid path for data to move directly between each CPU's cache.
- *Interprocessor communication*. An SMP OS requires communication between CPUs, which sometimes is best accomplished without accessing memory. Also, the system must often regulate interprocessor communication using a spin-lock that synchronizes access to a protected resource. Other SMP OS communication between CPUs is best accomplished without accessing memory. Systems frequently must also synchronize asynchronously. One such mechanism uses the device's interrupt system to cause activity on a remote processor. These software-initiated interprocessor interrupts (IPI) typically use an interrupt system designed to interface interrupts from I/O peripherals rather than another CPU.

Figure 5 shows how the ARM11 MPCore integrates the new ARM GIC inside the core to make the interrupt system's access and effects closer and more efficient. ARM designed the GIC to optimize the cost for the key forms of IPI used in an SMP OS.

Interrupt subsystem

A key example of IPI's use in SMP involves a multithreaded application that affects some state within the processor that is not hardware-coherent with the other processors on which the application process has threads running. This can occur when, for example, the application allocates some virtual memory. To maintain consistency, the OS

also must apply these memory translations to all other processors. In this example, the OS would typically apply the translation to its processor and then use the low-contention private peripheral bus to write to an interrupt control register in the GIC that causes an interrupt to all other processors. The other processors could then use this interrupt's ID to determine that they need to update their memory translation tables.

The GIC also uses various software-defined patterns to route interrupts to specific processors through the interrupt distributor. In addition to their dynamic load balancing of applications, SMP OSs often also dynamically balance the interrupt handler load. The OS can use the per-processor aliased control registers in the local private peripheral bus to rapidly change the destination CPU for any particular interrupt.

Another popular approach to interrupt distribution sends an interrupt to a defined group of processors. The MPCore views the first processor to accept the interrupt, typically the least loaded, as being best positioned to handle the interrupt. This flexible approach makes the GIC technology suitable across the range of ARM processors. This standardization, in turn, further simplifies how software interacts with an interrupt controller.

Snoop control unit

The MPCore's SCU is an intelligent control block used primarily to control data cache coherence between each attached processor. To limit the power consumption and performance impact from snooping into and manipulating each processor's cache on each memory update, the SCU keeps a duplicate copy of the physical address tag (pTag) for each cache line. Having this data available locally lets the SCU limit cache manipulations to processors that have cache lines in common.

The processor maintains cache coherence with an optimized version of the MESI (modified, exclusive, shared, invalid) protocol. With MESI, some common operations, such as A = A + 1, cause many state transitions when performed on shared data.

To help improve performance and further reduce the power overhead associated with maintaining coherence, the intelligence in the SCU monitors the system for a *migratory line*. If one processor has a modified line, and another processor reads then writes to it, the SCU assumes such a location will experience this same operation in the future. As this operation starts again, the SCU will automatically move the cache line directly to an invalid state rather than expending energy moving it first into the shared state. This optimization also causes the processor to transfer the cache line directly to the other processor without intervening external memory operations.

This ability to move shared data directly between processors provides a key feature that programmers can use to optimize their software. When defining data structures that processors will share, programmers should ensure appropriate alignment and packing of the structure so that line migration can occur. Also, if the programmers use a queue to distribute work items across processors, they should ensure that the queue is an appropriate length and width so that when the worker processor picks up the work item, it will transfer it again through this cache-to-cache transfer mechanism. To aid with this level of optimization, the MPCore includes hardware instrumentation for many operations within both the traditional L1 cache and the SCU.

The ARMv6K ISA can be considered a key multiprocessor-aware instruction set. With its foundation in low-power design, the architecture and its implementation in the ARM11 MPCore can bring low power to high-performance designs. These new designs show the potential to truly change how people access technology. With more than 1.5 billion ARM processors being sold each year, there is a huge range of markets in which ARM developers can use their software code. ■

References

1. D. Seal, *ARM Architecture Reference Manual*, 2nd ed., Addison-Wesley Professional, 2000.
2. A. Sloss et al., *ARM System Developer's Guide*, Morgan Kauffman, 2004.

John Goodacre *is a program manager at ARM with responsibility for multiprocessing. His interests include all aspects of both hardware and software in embedded multiprocessor designs. Goodacre received a BSc in computer science from the University of York. Contact him at john.goodacre@arm.com.*

Andrew N. Sloss *is a principle engineer at ARM. His research interests include exception handling methods, embedded systems, and operating system architecture. Sloss received a BSc in computer science from the University of Hertfordshire. He is also is a Chartered Engineer and a Fellow of the British Computer Society. Contact him at andrew.sloss@arm.com.*

Configurable Processors: A New Era in Chip Design

Configurable processors enable system-on-chip designers to leverage the benefits of nanometer silicon lithography with relatively little manual effort. These processors can achieve much higher performance than processors with conventional fixed-instruction-set architectures through the addition of custom-tailored execution units, registers, and register files as well as specialized communication interface ports.

Steve Leibson
James Kim
Tensilica

Microprocessor evolution can be broadly divided into three eras, each producing chips suited to their time. During the 1970s, microprocessors grew from 4-bit logic-replacement devices to 16- and 32-bit designs that paved the way for PCs and workstations. Explosive growth in 32-bit chips wiped out the minicomputer in the 1980s, which also saw the appearance of digital signal processors and other specialized architectures. Reduced-instruction-set computing dominated the 1990s; even stalwart complex-instruction-set computing cores such as the x86 evolved into disguised RISC architectures, and microprocessors became an integral part of mainframes and supercomputers.

Over the past three decades, the microprocessor has emerged as a fixed, stand-alone, reusable block created by highly skilled specialists. Because developing good, efficient microprocessor architectures can take years, many designers have come to regard them as monolithic entities subject to change only over long time periods and after careful consideration by an anointed few.

However, the rise of application-specific integrated circuit and system-on-chip (SoC) manufacturing technologies in the 1990s has laid the groundwork for a new, fourth era—that of post-RISC, configurable processors. Development tools are now advanced enough to allow any designer to tailor a microprocessor core for specific application tasks and to generate the processor's register transfer level (RTL) description plus all of the requisite software-development tools for that architecture in minutes, a shockingly brief time relative to the time spent designing processors and their associated development tools in prior eras.

Because of this ability to rapidly tailor processors for specific application tasks, configurable processors make excellent building blocks for SoC design, and developers can use them to quickly create functional blocks that might otherwise require months of manual labor to develop using handcrafted RTL descriptions. Consequently, various end products ranging from network routers to consumer electronics such as camcorders, printers, and video games already incorporate multiprocessor SoCs.

Two recent developments have further enmeshed configurable processors into SoC design:

- fully automated, application-specific instruction-set tailoring; and

0018-9162/05/$20.00 © 2005 IEEE

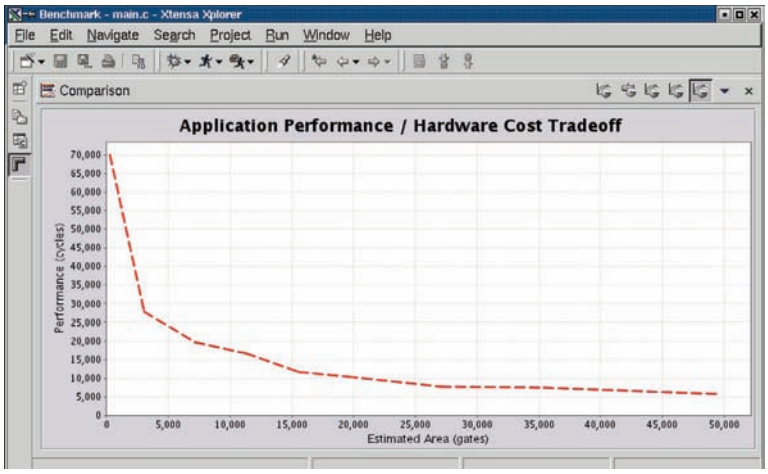


Figure 1. Xtensa processor extension synthesis compiler. XPRES creates a series of microprocessor configurations that provide increasing amounts of application-specific performance for an increasing amount of silicon area.

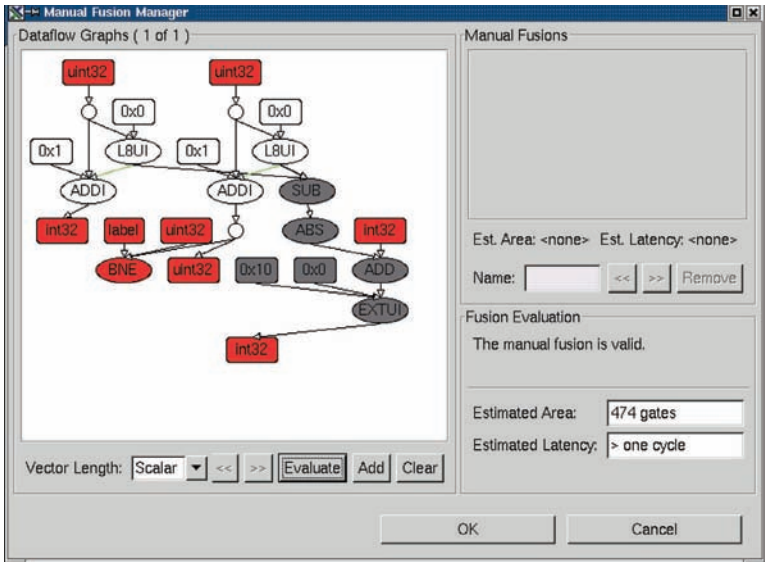


Figure 2. XPRES dataflow graph with a series of operations marked as fusible. In this example, EXPRES estimates that a new instruction that fuses the subtraction, absolute-value, addition, and bit-field-extraction operations will require 474 additional gates.

- multiport access to the processor's internal execution units.

With automated tools for tailoring processors, SoC designers can focus more on system architectural issues to achieve performance goals rather than spending a lot of time on designing individual functional blocks within the SoC. Multiport access permanently shatters the formerly ironclad bus bottleneck that has choked microprocessor performance since the first commercial chip appeared in 1971.

AUTOMATIC PROCESSOR TAILORING

For more than a decade, hardware designers have struggled to transform system specifications written in C, and later C++, into efficient hardware. Developers often use these languages to write initial system or application specifications because they can execute and evaluate the specifications on inexpensive PCs. However, even PC hardware is too costly for many embedded systems designs, especially in the consumer electronics arena. Designers have thus continued looking for a tool that reduces executable specifications written in C or C++ to hardware.

Various approaches—including behavioral synthesis, C-language hardware synthesis, and electronic system-level design—have all fallen short of the mark because they try to solve an essentially intractable problem: transforming a description written in a sequentially executable language into a parallel collection of interoperating, nonprogrammable hardware blocks.

Tensilica's Xtensa processor extension synthesis (XPRES) compiler uses a simpler, more direct approach to tackle this problem. Instead of attempting to create application-specific hardware from scratch, XPRES starts with a fully functional microprocessor core (Xtensa), which can already run any C or C++ program, and then adds hardware to it in the form of additional execution units and corresponding machine instructions to speed processor execution for the target application.

XPRES can search the available design space in less than an hour. This search results in a set of microprocessor configurations with a range of application performance and hardware cost characteristics (cost translates into silicon area on the SoC), as Figure 1 shows. The development team only needs to pick the configuration with the right performance/cost tradeoff for the target application and then submit it to Tensilica's Xtensa processor generator for implementation.

PERFORMANCE OPTIMIZATION

XPRES uses three techniques to create optimized Xtensa processor configurations: operator fusion; single instruction, multiple data (SIMD) vectorization; and flexible-length instruction extensions (FLIX).

Operator fusion

This technique notes the frequent occurrence of simple-operation sequences in program loops. XPRES combines these operation sequences into one enhanced instruction, which accelerates code execution by cutting the number of instructions executed within the loop, making the loop run faster, as well as reducing the number of instructions that must be fetched from memory, thus

decreasing bus traffic. Figure 2 shows an XPRES-generated operation dataflow graph, with fusible operations marked in gray.

SIMD vectorization

Many loops within application programs repetitively perform the same operations on an array of data items. To vectorize such loops, XPRES creates an instruction with multiple identical execution units that operate on multiple data items in parallel. XPRES automatically tries two-, four-, and eight-operation SIMD vectorization in its design-space exploration. The addition of SIMD instructions to an Xtensa processor dovetails with Tensilica's Xtensa C/C++ (XCC) compiler, which has the ability to unroll and vectorize application programs' inner loops.

The loop acceleration achieved through vectorization is usually on the order of the number of SIMD units within the enhanced instruction. Thus, a two-operation SIMD instruction approximately doubles loop performance, and an eight-operation SIMD instruction speeds up loop execution by about a factor of eight.

FLIX

Unlike the multiple dependent operations of fused and SIMD instructions, FLIX instructions consist of multiple independent operations. Tensilica's XCC compiler can pack these operations into a FLIX-format instruction as needed to accelerate code. While fused and SIMD instructions are 24 bits wide, FLIX instructions are either 32 or 64 bits wide to allow the flexibility needed to fully describe multiple independent operations.

MULTIPLE CONFIGURABLE PROCESSORS

Few applications today can achieve their performance goals with a single processor, even with a configurable processor tailored to the target application's needs. However, the multiprocessor instruction sets, high-bandwidth interfaces, and small size of configurable processors encourage their extensive use in SoC designs. Advanced SoCs commonly use 10 or more configurable processors, and some high-end SoC designs use more than 100 complete processors per chip.

The choice of hardware-interconnection mechanisms among processor blocks in a SoC greatly affects performance and silicon cost, and these mechanisms must directly support the system design's interconnection requirements. Message-passing software communications naturally correspond to data queues, but message passing can be implemented using other types of hardware such as bus-based devices with global memory. Similarly, the shared-memory software-communications mode naturally corresponds to bus-based hardware, but techniques exist to physically implement shared-memory protocols even when no globally accessible physical memory exists. This implementation flexibility lets chip designers implement a spectrum of different task-to-task connections to optimize performance, power, and cost.

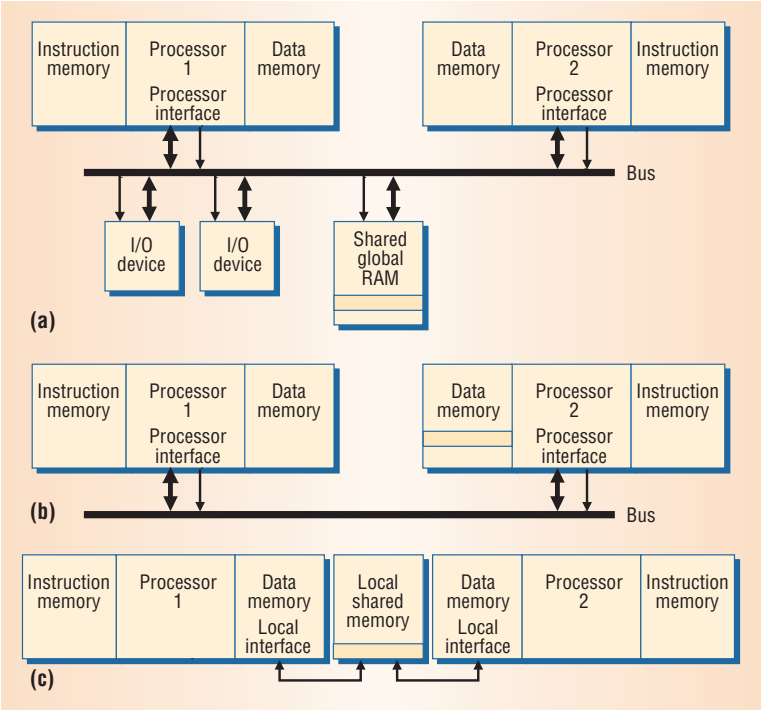


Figure 3. Shared-memory bus topologies using configurable processors. (a) Two processors access global memory over a bus. (b) One processor accesses the local data memory of a second processor over a bus. (c) Two processors share access to local data memory.

Configurable processors offer significant flexibility in supporting arbitrated access to shared devices and memory. The basic topologies for shared-memory buses are accessing remote global memory over a general processor bus, accessing local processor memory over a general processor bus, and accessing multiported local memory over a local bus.

Accessing global memory over a general bus

The processor can implement a general-purpose interface that allows a wide variety of bus transactions. If the processor determines that corresponding data is not local during a read, based on the target address or due to a cache miss, it must make a nonlocal reference over its main bus. The processor requests control of the bus, acquires bus control, and then sends the target read address over the bus. The appropriate device—for example, memory or an I/O interface—decodes that address and supplies the requested data back over the bus to the processor, as Figure 3a shows.

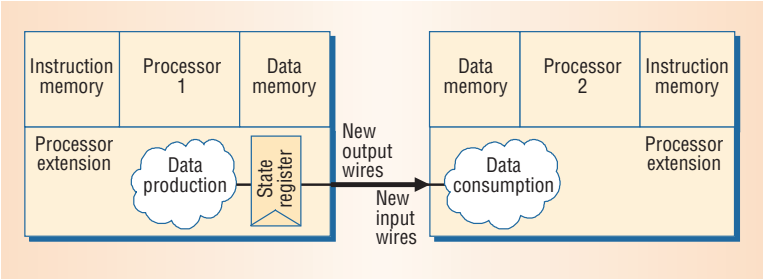


Figure 4. Direct processor-to-processor ports. Direct connection allows data to move directly from one processor's registers to the registers and execution units of another, reducing communication cost and latency.

When two processors communicate through global shared memory on the bus, one processor must acquire bus control to write the data; the other processor must later acquire bus control to read it. Each word transferred in this fashion requires two bus transactions.

This approach requires modest hardware and maintains high flexibility because the global memories and I/O interfaces are accessible over a common bus. However, using global memory is inefficient and does not scale well with the number of processors and devices because increased bus traffic leads to long and unpredictable contention latency.

Accessing local memory over a general bus

Configurable processors can allow local data memories to participate in general-purpose bus transactions. These data memories are primarily used by the processor to which they are closely coupled. However, the processor controlling the local data memory can serve as a bus slave and respond to requests on the general-purpose bus, as Figure 3b shows.

In this case, the read by processor 1 can require access arbitration when processor 1 requests access to the general-purpose bus and again when the read request reaches processor 2. The read request arrives over processor 2's interface and might contend with other requests for local-data-memory access from tasks running on processor 2. These two levels of arbitration can increase the access latency that processor 1 encounters, but processor 2 avoids access latency almost entirely because latency to local data memory is short—usually one or two cycles.

This latency asymmetry between processors 1 and 2 encourages *push communication*: When processor 1 sends data to processor 2, it writes the data over the bus into processor 2's local data memory. If the write is buffered, processor 1 can continue execution without waiting for the write to complete. Thus, the long latency of data transfer to processor 2 is hidden. Processor 2 sees minimal latency when it reads the data because the data is local. Similarly, when processor 2 wants to send data back to processor 1, it writes the data into processor 1's local data memory.

Accessing multiported local memory over a local bus

When data flows in both directions between processors and latency is critical, a locally shared data memory is often the best choice for intertask communications. Each processor uses its local-data-memory interface to access shared memory, as Figure 3c shows. This memory could have two physical access ports (which can handle two memory references each cycle), or it could be controlled by a simple arbiter that holds off one processor's access for a cycle while the other processor is using the single physical access port.

Arbitration for a single port is preferred in area- and cost-sensitive applications, especially when shared-memory utilization is modest, because a true dual-ported memory is about twice as big per bit as single-ported RAM. However, true dual-ported memory may be preferable when the shared memory is very small or when the application requires absolute determinism of access latency.

DIRECT-CONNECT PORTS

Direct processor-to-processor connections reduce communication cost and latency by allowing data to move directly from one processor's registers to the registers and execution units of another. Figure 4 shows a simple example of direct connection. This example exploits the exporting of register state and imported wire values—features found in some extensible processors—to create an additional dedicated interface within each processor and to directly connect them.

When processor 1 writes a value to the output register, usually as part of some computation, that value automatically appears on the processor's output port. That same value is immediately available as an input value to operations in processor 2. Wire connections can be arbitrarily wide, allowing the quick and easy transfer of large and non-power-of-two-sized operands.

The operation that produces data for the output state register can be a simple register-to-register transfer or a complex logic function based on many other processor state values. Similarly, the receiving processor can simply transfer the input value to a processor state (register or memory) within itself, or it could use the value as one input to a complex logic function.

DATA QUEUES

The highest-bandwidth mechanism for task-to-task communication is hardware implementation of data queues. One data queue can sustain data

rates as high as one transfer every cycle, or more than 10 Gbytes per second for wide operands (tens of bytes per operand at a clock rate of hundreds of MHz) because queue widths need not be tied to a processor's bus width or general-register width. The handshake between data producer and consumer is implicit in the interfaces between the processors and the queue's head and tail.

Push and pop operations

The data producer pushes the data into the tail of the queue, assuming the queue is not full; if the queue is full, the producer stalls. When ready, the data consumer pops data from the head of the queue, assuming the queue is not empty; if the queue is empty, the consumer stalls.

The SoC designer also can create nonblocking push and pop operations for queues. Such queue operations in the data producer explicitly check for a full queue before attempting a push. The data consumer can explicitly check for an empty queue before attempting a pop. These mechanisms let the data producer or consumer move to other work in lieu of stalling.

Application-specific processors' instruction-set extensions allow direct implementation of queues. An instruction can specify a queue as one of the destinations for result values or use an incoming queue value as one operand source. Such operations can create a new data value or use an incoming data value during each cycle on each queue interface.

As Figure 5 shows, a complex processor extension can perform multiple queue operations per cycle, combining input from two input queues with local data and sending values to two output queues. A queue's high aggregate bandwidth and low control overhead enable using application-specific processors for applications with very high data rates, which processors with conventional bus or memory interfaces cannot handle.

Queue sizing

Queues decouple the performance of one task from another. If the data production and consumption rates are uniform, the queue can be shallow. If either the production or consumption rate is highly variable, a deep queue can mask this mismatch and ensure throughput at the average data producer and consumer rates, rather than at their minimum rate.

Queue sizing is an important optimization driven by good system-level simulation. If the queue is too shallow, the processor at one end of the communication channel can stall when the other processor slows for some reason; if the queue is too deep, the silicon cost will be excessive.

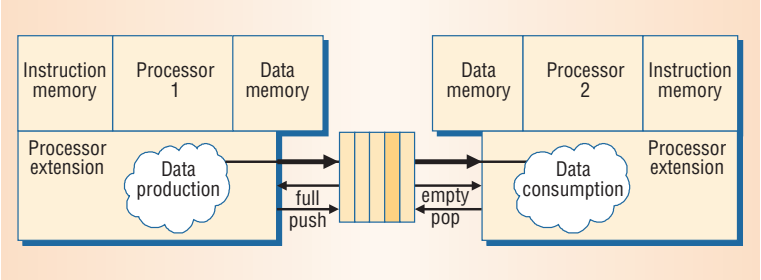


Figure 5. Data-queue mechanism. A complex processor extension can perform multiple queue operations per cycle, combining input from two input queues with local data and sending values to two output queues.

Queue interfaces

Queue interfaces to processor execution units are an unusual feature of commercial microprocessor cores. They become part of an Xtensa LX processor through the Tensilica instruction extension (TIE) language syntax, which defines the queue's name, width, and direction:

```
queue <queue-name> <width> inlout
```

One Xtensa LX processor can have more than 300 queue interfaces of variable width up to 1,024 bits each. These limits are set beyond the routing limits of current silicon technology so that the processor core's architecture is not the limiting factor in a system's design. The designer sets the practical limit based on system requirements, computer-aided design flow, and process technology selection. Using queues, designers can trade off fast and narrow processor interfaces with slower and wider interfaces to achieve bandwidth, performance, and power goals.

Figure 6 shows how TIE queues easily connect to simple DesignWare first-in, first-out memories. FIFO empty and full status signals gate TIE queue push and pop requests to comply with the DesignWare FIFO specification. The diagn_input is driven high, and the almost_full, half_full, almost_empty, and error outputs are unused. More elaborate FIFO memory implementations might be able to exploit the request signals when FIFO memory is nearly empty or full.

TIE queues serve directly as input and output operands of TIE instructions, just like a register operand, state, or memory interface. The following TIE syntax creates a new instruction that accumulates values from an input queue into a register file:

```
operation QACC {inout AR ACC} {in IQ1} {
        assign ACC = ACC + IQ1;
}
```

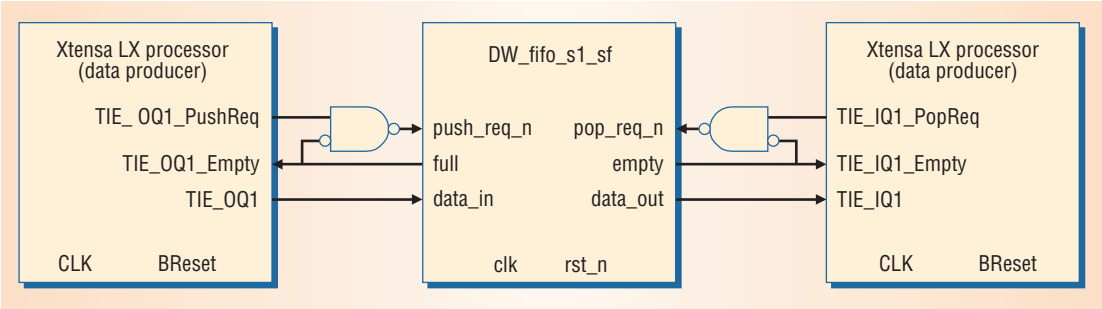


Figure 6. DesignWare synchronous first-in, first-out controller used with TIE queues. FIFO buffering provides a registered and synchronous interface to the external agent. Two entries buffer the processor from stalls when the attached external FIFO controller is full. In addition, buffering hides the processor's speculative execution from the external FIFO controller.

Figure 7 shows how TIE queues can function just like other instruction operands in an Xtensa LX processor. The figure also illustrates a key difference between queue interfaces and memory interfaces: The system designer can customize the width of each queue interface port to the exact value desired, either wider or narrower than the processor's standard memory interface ports.

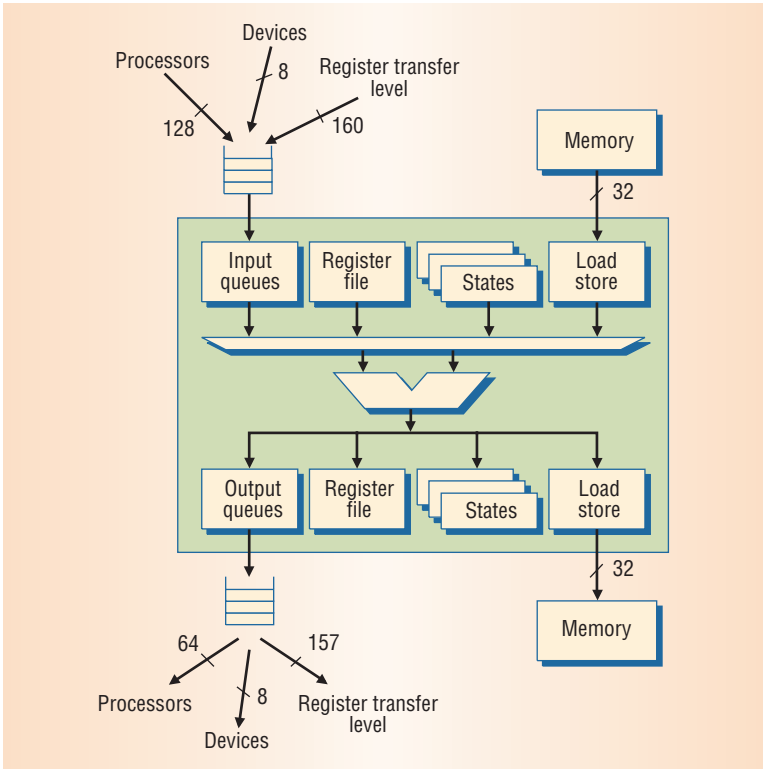


Figure 7. TIE queues used as instruction operands. The system designer can customize each queue interface port to the exact value desired, either wider or narrower than the processor's standard memory interface ports.

Queue buffering

Whereas memory accesses often exploit temporal locality, queue data is naturally transient. Consequently, queue storage can typically be smaller than a general-purpose memory buffer used for similar purposes. The Xtensa LX processor includes two-entry buffering for every TIE queue interface that the system designer defines. A queue interface's two-entry buffer consumes a substantially smaller area than a memory load/store unit, which can have large combinational blocks for alignment, rotation, and sign extension of data as well as cache-line buffers, write buffers, and complicated state machines. Thus, the processor area that TIE queue interface ports consume is under the designer's direct control and can be quite small or as large as necessary.

The FIFO buffering incorporated into the Xtensa LX processor for TIE queues serves three distinct purposes. First, it provides a registered and synchronous interface to the external agent (the actual FIFO memory), which portable IP blocks need to meet timing requirements under widely varying uses. Second, for output queues, buffering provides two entries that buffer the processor from stalls when the attached external FIFO memory is full. Third, it hides the processor's speculative instruction executions from the external FIFO memory.

HANDLING SPECULATION

Speculative loads occur on input queue interfaces because instructions operate on the queue data before these instructions are guaranteed to have completed all operations (before they reach the processor's commit stage). Activating a queue interface after the commit stage could be nonspeculative, but it would also be less useful, because a subsequent instruction that operated on that queue

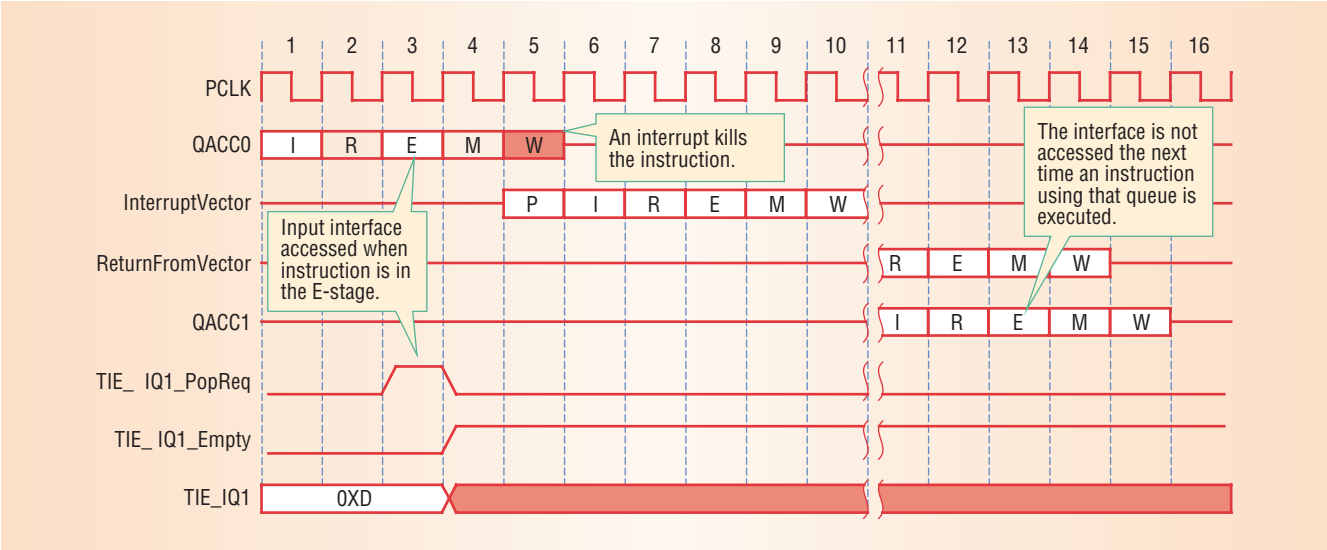


Figure 8. Speculative buffering. The processor handles speculative loads from a FIFO controller by keeping a temporary copy of data read from an input interface in a special buffer.

data would have to wait several cycles for the read-after-write hazard to resolve.

Speculative buffering

The processor can handle speculative loads from FIFO memory more effectively via *speculative buffering*, as Figure 8 shows. As an instruction reads data to be used as an operand from a TIE queue input interface, the queue's dedicated speculative buffer stores a copy of that data. The speculative buffer frees this entry only when the instruction commits. If the instruction does not commit—due to an exception, interrupt, or branch—the queue data remains in the buffer until the processor executes the next queue-reading instruction. This second execution obtains the internally buffered data, rather than reading a new value from the external FIFO memory, thereby preserving the ordering and coherence of queue references.

Figure 9 shows an example of speculative buffer timing. In cycle 3, the QACC0 instruction reads the input queue IQ1. Before the instruction can commit, an interrupt in cycle 5 kills it. The next instruction to execute that reads the input queue IQ1 is the QACC1 instruction. When this instruction executes, it uses the buffered queue data rather than issuing a new pop request to the associated FIFO memory in cycle 13.

Speculative writes to output queues

Speculative writes to output queues are much simpler and work similarly to speculative writes to the processor's register files and states. That is, writes to output queues are only visible outside of the processor when the instruction commits. Speculation within the processor is handled by pipelining results to the commit stage.

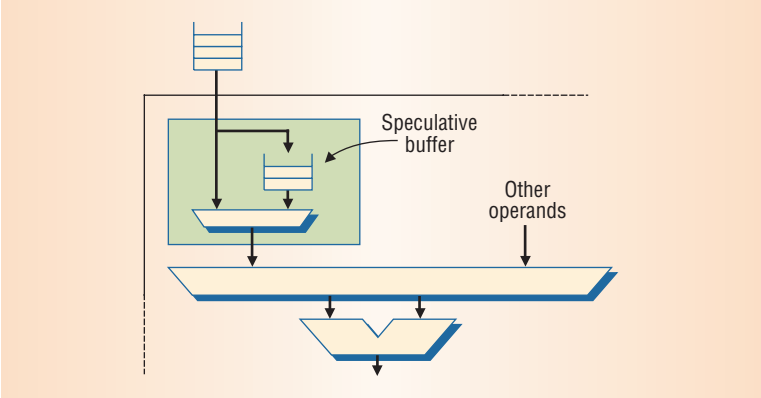


Figure 9. Example of speculative buffer timing. If the input queue indicates that it is not ready by asserting TIE_IQ1_Empty, by default, a queue is blocked, and processor execution stalls until it becomes available.

For example, in Figure 9, if the input queue indicates that it is not ready by asserting TIE_IQ1_Empty, by default, the queue is blocked and processor execution stalls until data becomes available. The same would be true for a write to a full output queue. This hardware blocking mechanism permits a simple and straightforward approach to synchronization between a producer and a consumer.

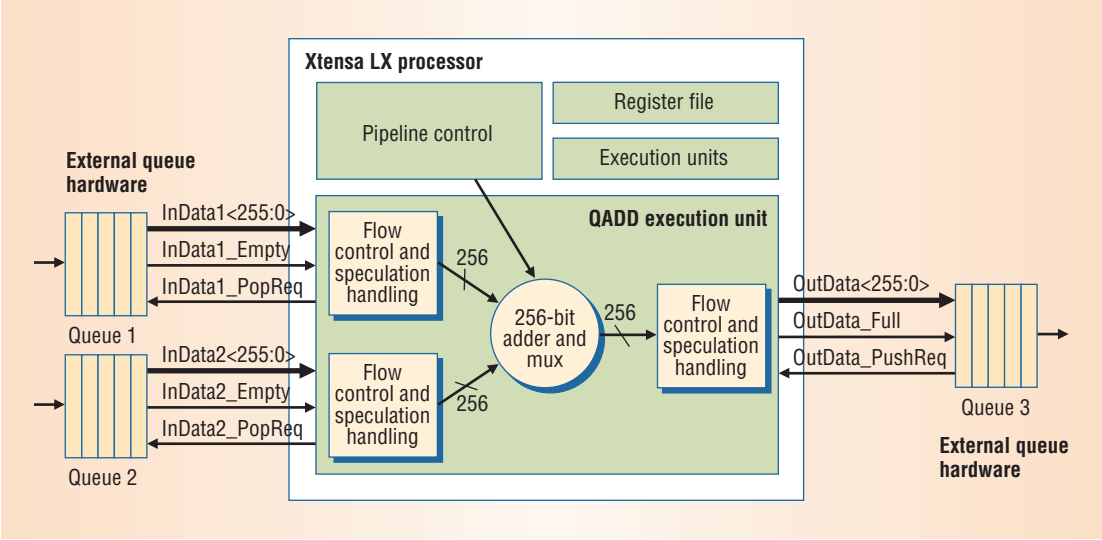


Figure 10. Flow-through processing. Combining queues with execution units adds flow-through processing to a configurable processor core.

In contrast, synchronizing communications using a shared-memory model is accomplished through semaphores and synchronization primitives, which is far more complicated. First, semaphores and data must use separate address spaces. The producer and consumer both poll the semaphore location to read each other's status. In addition, the data producer's synchronization software must guarantee the ordering of writes to the semaphore relative to writes to the data array to ensure the data consumer does not read the updated semaphore before all writes to data memory have completed.

There are many different approaches to memory ordering and synchronization, but all this effort is unnecessary when using a queue implementation that employs built-in hardware synchronization through FIFO memory's empty and full mechanisms.

Nonblocking queue accesses

Using nonblocking code for queue accesses is preferable when other tasks or processes can execute concurrently on the processor and when queue stalls can take many cycles. During these stalls it would be useful to switch to another task and return to the current thread later when the queue becomes available.

Code running on an Xtensa LX processor can perform nonblocking queue accesses by explicitly checking the queue's status and branching before executing the queue instruction itself, as shown in the following code snippets.

Assembly code for nonblocking queue access:

```
TASKA:
  check_queue_full b1  // queue status
    assigned to bool
  bnez b1, TASKB       // switch tasks if
    queue is full
  write_queue a1       // write to queue
        [...]
TASKB:
```

C for nonblocking queue access:

```
if(!check_queue_full()) {
      write_queue(value);
} else {
      [task b]
}
```

FLOW-THROUGH PROCESSING

The availability of ports and queues tied directly to a configurable processor's execution units permits the use of processors in an application domain previously reserved for hand-coded RTL logic blocks: *flow-through processing*. Combining input and output queue interfaces with designer-defined execution units makes it possible to create a firmware-controlled processing block within a processor that can read values from input queues, perform a computation on those values, and output the results with a pipelined throughput of one complete input-compute-output cycle per clock.

Figure 10 illustrates a simple design of such a system with two 256-bit input queues, one 256-bit output queue, and a 256-bit adder/multiplexer (mux) execution unit. Although this processor extension runs under firmware control, its operation bypasses the processor's memory buses and load/store unit to achieve hardware-like processing speeds.

Despite substantial hardware in this processor extension, its definition consumes only four lines of TIE code:

```
queue InData1 256 in
queue InData2 256 in
queue OutData 256 out
operation QADD {} { in InData1, in InData2,
  in SumCtrl, out OutData} { assign OutData
  = SumCtrl ? (InData1 + InData2) : InData1;
  }
```

The first three lines define the 256-bit input and output queues, and the fourth line defines a new processor instruction, QADD, which performs 256-bit additions or passes 256-bit data from input to output. Defining the instruction in TIE tells the Xtensa processor generator to automatically add the appropriate hardware to the processor and to add the new instruction to the processor's software-development tool set.

Fixed-core processors with a fixed instruction set and limited numbers of I/O ports and load/store units were appropriate in the days when microprocessors came in pin-limited packages, software development tools were handcrafted over a period of months or years, and system designs were undertaken at the board level. However, for 21st-century SoC design, such constraints are obsolete.

The configurable processor represents the next evolutionary step in microprocessor development, paving the way for many new and interesting system architectures that employ multiple, heterogeneous processor cores and exploit the qualities of advanced semiconductor lithography. Configurable processors provide SoC designers with building blocks that achieve performance rivaling hand-built RTL hardware blocks with postfabrication flexibility and firmware programmability but with much lower block-development and verification costs. ■

Steve Leibson *is technology evangelist for Tensilica Inc. (www.tensilica.com), based in Santa Clara, California. His research interests include processor architectures and advanced system-level design. Leibson received a BSEE from Case Western Reserve University. He is a senior member of the IEEE and a 30-year member of the IEEE Computer Society. Contact him at sleibson@tensilica.com.*

James Kim *is a senior design engineer in the hardware group at Tensilica Inc. His research interests include high-bandwidth processor interfaces and low-power design. Kim received an MSEE from Stanford University. Contact him at jameskim@tensilica.com.*

An Open Platform for Developing Multiprocessor SoCs

A low-cost modular approach that uses emulation offers an alternative to software simulation for the design and verification of complex multiprocessor system-on-chip (MPSoC) designs.

Mario Diaz Nava
Patrick Blouet
Philippe Teninge
Marcello Coppola
Tarek Ben-Ismail
STMicroelectronics-Grenoble

Samuel Picchiottino
Robin Wilson
STMicroelectronics-Crolles

Nanometer technologies integrate hundreds of millions of transistors in a single chip. Opportunities provided by these technologies, combined with the consolidation of platform-based design approaches,[1,2] the evolution toward multiprocessor architectures, and consideration of the network-on-chip (NoC) paradigm[3] suggest new methods for designing and verifying embedded systems.

Clearly, a pure software simulation platform can't provide the performance required for developing multiprocessor system-on-chip (MPSoC) designs. What's more, one of the main design risks for today's systems is the architecture, which developers must validate as early as possible in the overall system design cycle because it has the biggest impact on system dimensioning and performance.

Our approach introduces concurrent hardware and software engineering early in the development process, and uses low-cost emulation facilities.

We also extend the emulation used for verifying application-specific integrated circuits (ASICs) and application-specific standard product (ASSP) devices to multiprocessor architectures.

Several multiprocessor emulation platforms, such as the ARM Integrator (www.arm.com) and Hunt Engineering's Heron (www.hunteng.co.uk), already exist on the market. However, these platforms use proprietary processors and interconnects and don't allow the addition of third-party IPs such as memory controllers.

In general, existing solutions can't evaluate different architectures because their interconnect topologies can't be modified. Additionally, these platforms primarily perform functional validation of a given application or design, such as a prototype. Although this approach is useful, it doesn't evaluate and verify the architecture, which will in fact be implemented on silicon.

To solve these problems, we've studied a reconfigurable MPSoC emulation platform and developed the main emulation subsystem and board. We use this platform to design and verify a specific range of products based on in-house cores and interconnects. We've designed the platform for use in different modes:

- stand-alone for software development and intellectual property (IP) design and verification,
- as a multiprocessor system allowing its direct connection to a PC, and
- as an MPSoC computing node or tile for reconfiguring the interconnect to evaluate the best-performing communication system and emulating a NoC topology.

We plan to introduce this platform in consumer and telecommunications product development to increase software and hardware engineers' productivity. This will reduce development time and costs while ensuring design and product quality.

EMULATION AND DESIGN METHODOLOGIES

An emulation platform is a hardware system used to map a hardware design. It can run at one- to

Published by the IEEE Computer Society

several-MHz operation frequency and can include software execution. A prototyping board is like an emulation platform but devoted to a specific application. It has limited debug capabilities and operates at higher speeds than hardware emulators. In general, emulation platforms are reconfigurable and more flexible than prototyping platforms.

These platforms are generally built on field-programmable gate arrays (FPGAs) and specific chips to ensure that the chips interconnect. They range from simple, for systems devoted to validating IP blocks, to complex, for systems allowing the validation of a full SoC, which can cost from $50,000 to around $1 million.

As the "Existing Emulation Platforms" sidebar indicates, we studied several current emulation systems but found that none of them were adequate for our design purposes. The more flexible systems—the hardware emulators—are expensive and slow and don't provide the tools software developers require. Although the ARM Integrator is more suitable for software development, it isn't open enough for nonproprietary ARM-based IP blocks.

We therefore developed our own system, driven by two key requirements:

- It must use both proprietary and third-party cores and interconnects.
- It must be inexpensive, flexible, modular, and multimodal.

Mastering design complexity, reducing development time, and improving design quality requires combining system-level design methodologies with hardware emulation capabilities.

Nobuyuki Ohba and Kohji Takano described an interesting design methodology starting at the system level,[4] which we use to develop and verify IP blocks.

Mohamed-Wassim Youssef and colleagues proposed a methodology for designing an MPSoC system[5] that we also apply in our open platform.

For more extensive verification, hardware designers must provide seamless flows (and associated tools) with which to synthesize designs from C/C++ descriptions into register transfer level and then into a gate-level netlist, mapping the results in hardware emulators. This flow will increase design productivity and quality.

ARCHIFLEX SYSTEM PLATFORM

We had two main goals for our system. First, we wanted an open and reconfigurable hardware emulation platform matching STMicroelectronics'

Existing Emulation Platforms

Emulation and Verification Engineering (EVE, www.eve-team.com) designed its system emulator as a single board using several field-programmable gate arrays (ZeBu-ZV model) to validate intellectual property (IP) blocks with cosimulation capabilities. That is, the board connects to a simulator at the cycle and transactional levels.

The system's cost, capacity, high speed, and cosimulation capabilities provide an attractive solution for hardware and software design verification and validation. EVE recently increased the ZeBu-XL's emulation capacity from 1.5 million gates (provided by ZeBu-ZV) to 48 M gates. The cost, which varies from $50 to $120,000, is proportional to the design's complexity.

Aptix's System Explorer MP4 (www.aptix.com) is built around a complex motherboard with several FPGAs and field programmable interconnect (FPIC) technology to interconnect the different chips. Developers use an extension module plugged into the motherboard to add existing chips as processors, digital signal processing (DSP), and so on.

For several years, designers have used MP4 as an emulation platform for prototyping complex designs (up to around 5 Mgates) to validate hardware and software prior to silicon—that is, before sending the design to fabrication—and find bugs they can fix in silicon, rather than patch the software at a later date. However, the system is too expensive for wide deployment.

Cadence's Palladium (www.cadence.com) is the newest generation of hardware emulators. Palladium II is built around a massively parallel Boolean computing engine and processor-based architecture. Its basic building block is a multichip module (MCM), which includes a multiprocessor array chip and memory chips. Several MCMs are arrayed on boards to provide a 256-Mgate accelerator/emulator capacity.

Palladium can map very complex systems (both hardware and software) and provides a high level of design observability. Its drawbacks are its cost (around $1 million) and operation frequency (around 1.6 MHz). Palladium is so expensive that it's not suitable for concurrent design development; rather, it's most useful for validating a hardware or software system's correct operation prior to silicon.

ARM's Integrator/ASIC development platform (AP, www.arm.com) supports up to five compute nodes (processor plus memory) and an FPGA board. The advanced microcontroller bus architecture (AMBA) connects all of the boards and has a bridge to adapt the AMBA protocol for use by the peripheral component interconnect (PCI).

Because of its low FPGA capacity, the first ARM Integrator solution could only map low-complexity hardware. Subsequent versions improved this capacity. Designers have used the platform to prototype systems based on the ARM core and to validate system functions with the limitation that such a system doesn't necessarily represent the architecture that will be implemented on silicon.

We studied the Integrator/AP with the goal of replacing the AMBA interconnect with another interconnect, but we encountered several limitations: few available interconnect signals, the low capacity of the FPGA used as a network interface between the ARM tile and the interconnect, and the impossibility of changing the FPGA code because it's ARM proprietary.

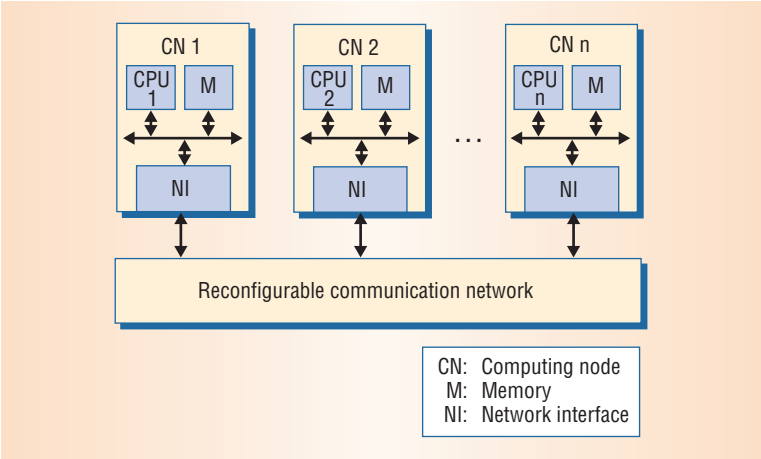


Figure 1. Heterogeneous multiprocessor system. The architecture's three main components are the computing nodes (CNs), network interface (NI), and interconnect. The CPU accesses its local memory (M) through its local bus.

needs in terms of interconnect (architecture, protocol stack, connectivity resources, and so on) and a heterogeneous multiprocessor environment.

Second, we sought to develop a modular software design methodology using a layered approach so that developers could write application software independent of the hardware architecture. We wanted to isolate processors' and peripherals' communication actions (closely related to the interconnect architecture and the final system performance) from their computation actions.

With this approach, we could migrate the software from one architecture to another with minimal effort. This implies that developers could change the interconnect and its topology with little effect on the software architecture, allowing them to evaluate and explore different architectures. The approach also provides tools for use in defining and dimensioning MPSoC systems.

As Figure 1 shows, the Archiflex emulation platform's architecture is based on a nonuniform memory access (NUMA)[6] scalable shared-memory multiprocessor organization.

The architecture consists of three main components:

- The *computing node* (or tile) is representative of the platform-based design of a range of in-house multimedia products. It includes an ST230 processor core member of the very long instruction word (VLIW) family, different cache memory levels, and input/output ports with direct memory access (DMA) capabilities. An STBus interconnect connects the components.[7]
- The *network interface* connects the computing node to the communication network or the system interconnect. It includes a reprogrammable FPGA to support different protocols and interconnect interfaces.
- The *interconnect* supports interconnect topologies of varying complexity. Its flexibility should allow developers to evaluate different topologies and communications schemes and identify those that best fit the system requirements. Trial results will more accurately reflect communication performance because of the high traffic volume this type of emulator handles.

In addition, hardware and software engineers can use the platform in three modes:

- stand-alone mode, to develop and debug hardware IP and programs and efficiently perform low-cost hardware/software codesign from their desks.
- connection to a host PC, which software engineers can use to develop firmware, middleware, or application software on a PC and then reuse the PC hardware for I/Os. Our system provides a direct connection interface with the PC and a remote-access interface.
- MPSoC mode, in which the platform supports up to eight computing nodes.

Our modular approach uses two types of boards: the computing node board with network interface capabilities and the interconnect board. A set of tools and hardware capabilities for developing and debugging software and hardware completes the open platform.

PLATFORM IMPLEMENTATION

Performance, reuse, and cost drove the implementation phase of the Archiflex platform. We designed and implemented the computing node board using a VLIW-based SoC, 128-Mbyte double data-rate (DDR) memory, a Xilinx Virtex-2 FPGA to support the network interface function, and flash and SDRAM controllers. We're designing another communication network board to support a flexible interconnect based on FPGA devices.

Computing node board

Figure 2a is a block diagram of the computing node board and its associated SoC (we used the TC4SoC).[8] Figure 2b shows the TC4SoC block diagram, and Table 1 summarizes the main TC4SoC system features.

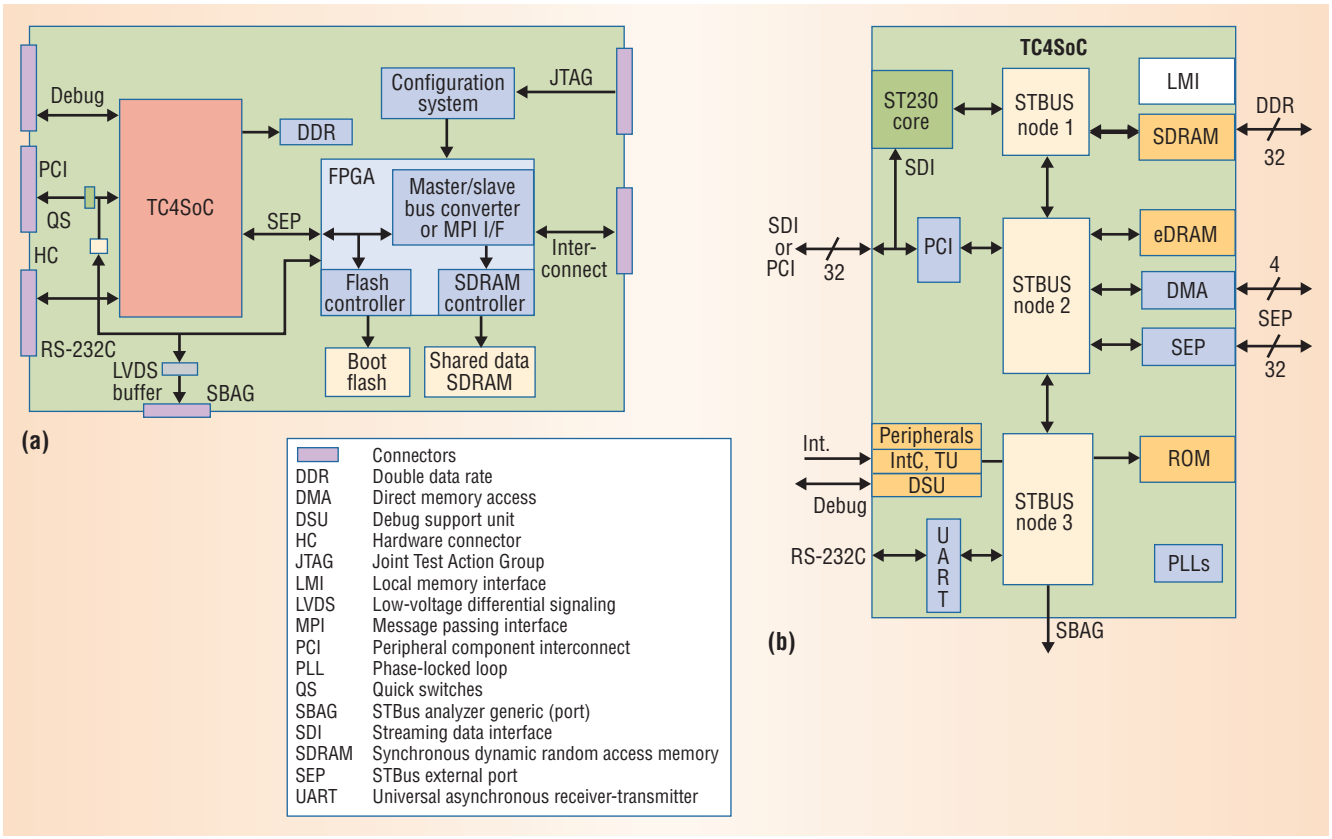


Figure 2. Block diagrams showing: (a) computing node board, which includes a microcomputer with additional external memory and programmable logic and communication programmable capabilities, and (b) TC4SoC. Using hierarchical interconnect segments (STBus nodes 1 to 3) improves the processor's performance by splitting the data flows between blocks to reduce bus contentions and allowing parallel exchanges.

To improve the processing node's performance, the TC4SoC architecture uses hierarchical interconnect segments (nodes 1 to 3 in Figure 2b). The VLIW core has a private node segment supporting its local memory, whereas a shared memory on a shared node segment supports data exchange with external processing nodes.

TC4SoC provides the following interfaces:

- four streaming data interface (SDI) channels (two inputs and two outputs), each 8 bits wide, streaming very high-bandwidth data through the processor and avoiding memory bus traffic and cache pollution;
- a local memory interface DDR SDRAM operating at up to 133 MHz (0.8 Gbytes-per-second peak throughput); and
- a 32-bit peripheral component interconnect (PCI) interface operating at 66/33 MHz.

To keep timing coherency between the silicon version and its associated emulation platform, the internal STBus interconnect extends through the external STBus port. This, together with FPGA programmable logic, gives developers the flexibility to adapt the interface and protocol associated with this port to any other interconnect. Developers can configure this port as an initiator or target inter-

Table 1. TC4SoC main features.

Block	Description
32-bit very long instruction word (VLIW) CPU	• 400-MHz processor clock with four issues per cycle, providing up to 1.6 giga operations per second (GOPS) • 32-Kbyte-cache (direct mapped) and 32-Kbyte *D*-cache (four-way set associative) memories • Up to 128-bit-wide instruction word • Streaming data interface allowing up to 1.6 Gbyte-per-second data rates • Virtual memory support via a translation look-aside buffer
Direct memory access (DMA)	High-performance five-channel DMA engine supporting 1D or 2D block moves and scatter/gather
Interconnect	32-bit STBus; high-throughput, low-latency, split-transaction packet router; 133-MHz clock
Peripherals	• Interrupt controller with up to 64 interrupt sources • Debug support unit (DSU) • 32 × 32-bit programmable timer units • Two universal asynchronous receiver-transmitters
Embedded memory	• 512-Kbyte ROM • 256-Kbyte RAM • 1-Mbyte eDRAM
Debug	Port providing debug link interface to core

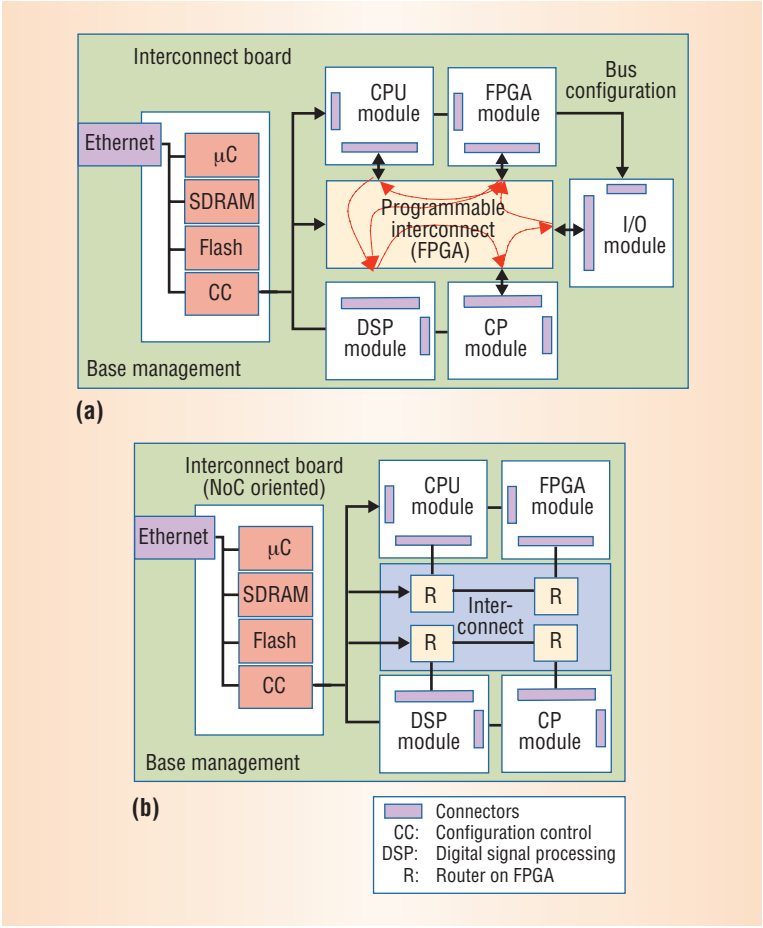


Figure 3. Interconnect boards for (a) systems with four or fewer computing nodes and (b) more complex MPSoC platforms. The board in (a) uses one FPGA with many input/output (I/O) pins; the interconnect board in (b) uses an array of FPGAs.

face. The port supports a 32-bit-wide bus and operates at up to 133 MHz. A debug port provides a debug link interface to the core.

The implemented FPGA provides flexibility to the complete computing node board. Developers can program this FPGA and use it as a network interface in MPSoC operation mode.

The FPGA's hardware programmability supports different protocols and interconnects. In standalone mode, the FPGA logic implements IP hardware blocks. The FPGA also implements the flash and SDRAM memory controllers.

The FPGA network interface implements the communication protocol required to interconnect the processing node and the other Archiflex nodes. The architecture uses the message-passing interface (MPI, www-unix.mcs.anl.gov/mpi) to share data and synchronize communication between nodes.

Unlike the solution proposed by Sang-Il Han and colleagues,[9] which separates control and data ports, Archiflex uses in-band control and multiplexes the different channels in a single port per computing node. Thus, the same network interface can support several ports.

Communication network board

The communication network board ensures communication between computing nodes. We considered two solutions. First, when there are four or fewer computing nodes, our design uses a single FPGA with more than 1,000 I/O pins, as Figure 3a illustrates. We mapped the interconnect in such an FPGA.

For more complex MPSoC platforms, our design uses an interconnect board oriented to a NoC. The board is based on an array of FPGAs, as Figure 3b shows. The platform's complexity is directly related to array size. This solution paves the way for a NoC topology.

Both boards include hardware and software facilities for configuring and managing the interconnect system.

Clock strategy

We carefully analyzed the clock issues for the computing node board. The TC4SoC architecture can use different frequencies to operate the internal blocks and the different interfaces (SDI, PCI, STBus external port, and DDR). For example, SEP can operate at 33, 66, 100, or 133 MHz. Nominal operation is 100 MHz.

STBus building blocks include clock, data, and protocol converters. For synchronization purposes, each interface provides a clock and can receive a clock. Software selects which clock will perform the synchronization.

For the overall platform, we used the following strategy to solve the clock domain issue. The network interface block (in FPGA) ensures synchronization between clock domains (interconnect, network interface, and computing nodes) via asynchronous first-in, first-out order using the FPGA's dual-port memory. If the clock shift is strong, the FPGA's internal phase-locked loop regenerates the incoming clocks.

Software tool chain

In addition to the hardware platform, the VLIW development system provides a set of tools for creating programs for the family of VLIW cores:

- The *simulator* evaluates code before running it on the target machine.
- The *compiler* generates the executable code to run on the target machine.
- *Runtime* provides low-level services for controlling key peripherals such as the timer, interrupt controller, and clock generator on the target machine and allows a VLIW program to

load and execute. While the program is executing on the target machine, runtime emulates some system calls on the host machine. Runtime's hierarchical structure makes it easily adaptable to various core/SoC/board combinations.
- The provided *debugger* is a VLIW retargeting of the GNU gdb standard debugger.
- The *operating system* lets developers write applications under an open OS such as Linux.

The environment is cross-developmental because the target machine for developing the code (for instance, ST230) is different from the host machine that runs the development tools.

Software and hardware debug facilities

The debug support unit debugs ST230 code and hardware from the host by providing direct access to the ST230 core. The DSU exchanges information with the host via a hardware debug probe. A Joint Test Action Group (JTAG) interface connects the DSU to the hardware debug probe. Ethernet connects the other side of the probe to the host.

The FPGA configuration code is loaded into the flash memory through a JTAG interface. The Xilinx (www.xilinx.com) Chipscope tool uses the JTAG port as a debugger to observe the internal signals in the FPGA. Unused FPGA outputs provide additional test points to track internal signals with an external logic analyzer. Hardware facilities in the TC4SoC track and store transactions performed on the internal STBus. The information sampled is available via the STBus analyzer generic (SBAG) port. The software analyzes the traffic and activity of each component connected to the STBus for debug and performance measurement purposes.

ARCHIFLEX IMPACT IN AN MPSOC APPLICATION

The SoCs in consumer applications are becoming increasingly complex. A DVD recorder, for example, requires a huge amount of processing power to support all of the features end users expect. A single CPU can't provide real-time support for all of the functions involved in running a DVD recorder application without hardware accelerators. Achieving the flexibility needed to handle new standards and system updates requires a multiprocessor system.

Archiflex supports a four-processor system that can handle such applications;

- The *host processor* manages the entire system, including the user interface dialog and the other three processors. This processor is based on a 266-MHz CPU and supports functions such as DVD playback, authoring, and recording; user control; and interface control.
- The *video encoding processor* is an ST230 running at 400 MHz. It supports and controls all video-encoding functions such as MPEG2. It benefits from some hardware assistance for consuming processing tasks.
- The *audio recorder* is a 400-MHz ST230. The recorder supports all of the major and standard audio codec used in consumer applications, requiring no additional hardware.
- The *audio decoder*, also a 400-MHz ST230 processor, supports audio-decoding tasks such as MP3.

In the cross-developmental environment, the target machine for developing the code is different from the host machine that runs the development tools.

Such a system clearly doesn't support a simulation platform with acceptable performance for software development. Even the most up-to-date workstations can't simulate very fast processors with high enough performance to allow the development of software components with an acceptable verification level (running a few frames of video and audio signals can take days of simulation).

Before developing the Archiflex architecture, we used a different approach at the start of the DVD project.

Initially, we used an early instance of the VLIW architecture to build a discrete prototype chip. We based this on the ST210 chip, which had a PCI interface onboard. We used the PCI interface to build up a DVD subsystem with a CPU chip that also had a PCI interface and three ST210-based chips. Using the PCI interface facilitated reusing some existing hardware for I/Os, such as audio or video boards.

Unfortunately, the prototype is far from what the final silicon will be. Even if suitable for functional software verification, it is inadequate for more accurate performance verification because the PCI protocol adds substantial overhead to the interprocessor communications. It also needs additional development because a PCI driver must be available on all of the platforms to support interprocessor exchange and synchronization.

Although helpful at the beginning of the project, this approach demonstrated some strong limitations that our open platform has addressed.

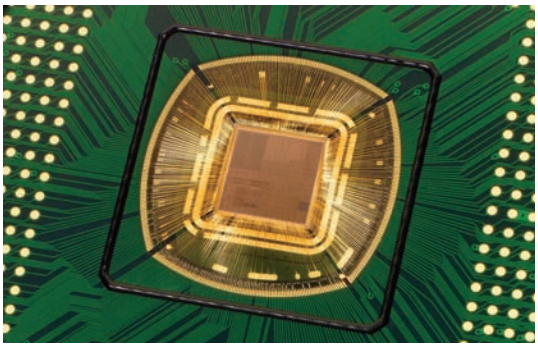


Figure 4. TC4SoC microphotograph. It shows a microcomputer chip, which includes a 32-bit VLIW core with its peripherals and input/output ports, and embedded memories of 1 Mbyte DRAM, 256 Mbytes SRAM, and 512 Mbytes ROM. The chip complexity is 40 Mtransistors in 35 mm².

An open multiprocessor SoC hardware platform's low cost would allow its wide adoption by software and hardware engineers in charge of designing and debugging future VLIW-based MPSoC products that include an STBus interconnect, advanced microcontroller bus architecture (AMBA), and other types of interconnects.

Archiflex supports the MPSoC's high design complexity, allowing concurrent hardware/software codesign from the earliest development stages. Our solution will also reduce development time and design and verification costs, leading to improvements in design quality.

Figure 4 shows the design and fabrication of the TC4SoC device in a 90-nanometer complementary metal-oxide semiconductor (CMOS) chip. The 35-mm² device is available in a 40 × 40-mm 520-pin ball grid array package. We've mounted the device in the computing node board; researchers are currently designing the interconnect board.

A global solution will let developers construct MPSoCs with up to eight computing nodes and a flexible interconnect for developing and validating IP and hardware/software codesign methodologies, evaluating architectures such as message passing and shared memory, and evaluating communication NoCs. It will also let them define an architecture based on NoC topology that can integrate 65-nm CMOS technology and serve as a platform-based design for next-generation multimedia products. ■

Acknowledgments

We thank Jean-Guy Bonneault for his participation in the design and boards specifications; Sylvan Engels and Benoit Foret for their contribution to the IC and system design; Michel Cazal and Paul Ghaleb for their software tool support; and Philippe Magarshack and Matthew Hatch for their support in the overall project development.

References

1. A. Sangiovanni-Vincentelli et al., "Benefits and Challenges for Platform-Based Design," *Proc. 41st Design Automation Conf.*, IEEE Press, 2004, pp. 409-414.
2. W. Cesario et al., "Multiprocessor SoC Platforms: A Component-Based Design Approach," *IEEE Design and Test of Computers*, Nov.-Dec. 2002, pp. 52-63.
3. L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, Jan. 2002, pp. 70-78.
4. N. Ohba and K. Takano, "An SoC Design Methodology Using FPGAs and Embedded Microprocessors," *Proc. 41st Design Automation Conf.*, IEEE Press, 2004, pp. 747-752.
5. M.W. Youssef et al., "Debugging HW/Software Interface for MPSoC: Video Encoder System Design Case Study," *Proc. 41st Design Automation Conf.*, IEEE Press, 2004, pp. 908-913.
6. D. Culler, J.P. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann, 1999.
7. STMicroelectronics, "STBus Functional Specifications," Apr. 2003, http://mcu.st.com/mcu/inchtml.php?fdir=pages&fnam=StBus_intro.
8. S. Picchiottino, R. Wilson, and M. Diaz Nava, "Platform to Validate System on Chip Design Flows and Methodologies," *Proc. IP-Based SoC Design Conf.*, Design and Reuse, Grenoble, France, 2004, pp. 39-44.
9. S-I. Han et al., "An Efficient Scalable and Flexible Data Transfer Architecture for a Multiprocessor SoC with Massive Distributed Memory," *Proc. 41st Design Automation Conf.*, IEEE Press, 2004, pp. 250-255.

***Mario Diaz Nava** is a system architect manager at STMicroelectronics-Grenoble. His research interests include digital communication circuits, multiprocessor systems, and system-level design methodologies. Diaz Nava received a PhD in computer science from the National Polytechnic Institute of Grenoble, France. He is a member of the IEEE. Contact him at mario.diaznava@st.com.*

***Patrick Blouet** is ST200 VLIW development tools manager at STMicroelectronics-Grenoble. His research interests include multiprocessor systems as well as methodologies related to the development of such systems. Blouet received an engineer*

degree in computer science from ENSERB. Contact him at patrick.blouet@st.com.

***Philippe Teninge** is an application lab engineer at STMicroelectronics-Grenoble. His research interests include design flow, FPGA prototyping, and system design. Teninge received an engineering degree in electronics, electro-techniques, and automatism from IST Grenoble. Contact him at philippe.teninge@st.com.*

***Marcello Coppola** is the head of the Grenoble research lab within the Advanced System Technology department at STMicroelectronics-Grenoble. His research interests include discrete event simulation, NoC, and MPSoC modeling and architecture. Coppola received a Laurea degree in computer science from the University of Pisa, Italy. Contact him at marcello.coppola@st.com.*

***Tarek Ben-Ismail** is a manager of the open platform program at STMicroelectronics-Grenoble. His research interests include technologies for multiprocessor systems, hardware/software codesign, and software tools for embedded applications. Ben-Ismail received a PhD in computer science from the National Polytechnic Institute of Grenoble. Contact him at tarek.ben-ismail@st.com.*

***Samuel Picchiottino** is a project leader at STMicroelectronics-Crolles. His research interests include on-chip communication, SoC architecture, design, and verification. Picchiottino received an MS in electrical engineering from the National Polytechnic Institute of Grenoble. Contact him at samuel.picchiottino@st.com.*

***Robin Wilson** is a design department manager at STMicroelectronics-Crolles. His research interests include design qualification of advanced CMOS processes, statistical-based design, and logic diagnostics. Wilson received a BEng degree from University College Cork, Ireland. He is a member of the IEEE. Contact him at robin.wilson@st.com.*

Evaluating Digital Entertainment System Performance

The Embedded Microprocessor Benchmark Consortium's DENBench suite of digital media benchmarks provides a spectrum of tools for assessing and refining the video and audio performance of digital devices.

Markus Levy
EEMBC

Digital entertainment systems—which encompass a broad range of applications, including smart phones, set-top boxes, MP3 players, DVD players and recorders, game consoles, and digital cameras—have become the driving force behind the expansion of the semiconductor market, outstripping even PCs. Consider, for example, that in 2003, smartphones represented about 3 percent of the 500 million mobile phones sold worldwide, with analysts expecting them to grow at triple-digit year-over-year rates.

Moving beyond the voice-centric model, more than half of the 600 million mobile phones sold in 2004 included a color display and digital camera. The implementation of more advanced features such as accelerated 2D and 3D graphics, videoconferencing, mobile multimedia, and games has raised the performance requirements of these models. The same holds true for any of the other digital entertainment devices.

Rapid advances in semiconductor technology, microarchitectures, and embedded systems have made the adoption of these features possible. As a result, software complexity will continue to increase to keep pace with the overall system complexities.

Manufacturers of devices such as semiconductors and entire systems race to create products with the latest technology and bring them to market before their competitors. Hence, the need to evaluate performance goes well beyond the consumer's demands for more entertainment and functionality value. Benchmarks and performance evaluation play an integral role in processor, algorithm, and system design.

CATEGORIZING DIGITAL ENTERTAINMENT BENCHMARKS

The best digital entertainment benchmark is the exact application that will be run on the device. For example, if the device will be running video playback, the appropriate test is to run an MPEG-4 decode algorithm. The more realistic the benchmark, the more valuable it will be for evaluation purposes. However, many modern benchmarks are very long, and performing simulations on them can become prohibitively time-consuming, as would be the case when evaluating IP cores or simulated platforms.

Research has shown that short synthetic streams of instructions can be created to approximately match the behavior of the instruction stream from the full execution.[1,2] However, these streams may not be sufficient to take system-level features such as caches and memory into account, and they certainly are not sufficient to test hardware accelerators and other CPU offload engines.

The first step in creating a suite of digital entertainment benchmarks is to define the target, since the benchmarks will differ depending on whether they are used to evaluate a microprocessor's microarchitecture, its system-level functionality, or an algorithm's behavior. Many proprietary benchmarks, such as BDTI and FutureMark, will fulfill these goals, but the approaches taken by university researchers and industry consortiums, such as MediaBench[3] and EEMBC (www.eembc.org), provide a more open and democratic alternative.

Published by the IEEE Computer Society
0018-9162/05/$20.00 © 2005 IEEE

MediaBench: An academic project

The MediaBench benchmark suite consists of several applications in the image processing, communications, and DSP categories. MediaBench includes applications such as JPEG (image compression), MPEG (video transmission encoding and decoding), GSM (full-rate speech transcoding), G.721 (voice compression), Ghostscript (PostScript language interpretation), and adaptive differential pulse code modulation (ADPCM).

An academic effort to assemble several media-processing-related benchmarks, MediaBench focuses on complete applications, using publicly available code, for multimedia and communications systems. The benchmark suite also requires the use of high-level language to stress compilation technology.

EEMBC: The consortium approach

The Embedded Microprocessor Benchmark Consortium's members formed this nonprofit organization in April 1997 to develop performance benchmarks for processors in embedded applications. Developers designed the original EEMBC (pronounced "embassy") benchmark suite to reflect real-world applications and some synthetic benchmarks. These benchmarks target the automotive and industrial, consumer, networking, office automation, and telecommunications markets.

EEMBC dissected applications from these domains and derived 37 individual algorithms that constitute its version 1.0 suite of benchmarks. In addition to developing the benchmark code, EEMBC seeks to provide certified benchmark scores to interested public sectors such as the embedded system design community.

EEMBC has evolved in parallel with the industry. The organization recently released a new suite of digital media benchmarks that include MP3 decode, MPEG-4 video encode and decode, MPEG-2 video encode and decode, image filtering, JPEG compression and decompression, and a variety of cryptography algorithms, as the "DENBench Suite" sidebar describes. Referred to as the DENBench suite for digital entertainment, this collection provides representative benchmarks for the audio, video, and still-image processing found in consumer electronics and home entertainment systems, videoconferencing systems, multimedia-enabled cellular telephones, and digital cameras.

Compared with previous EEMBC benchmark kernels that measure processor performance in consumer applications, the DENBench suite makes more intensive use of processor computational ability, caches, and system memory, thereby providing a detailed analysis of processor strengths and weaknesses. Going beyond evaluation of the processor core, the benchmarks also evaluate complex architectures and complex memory hierarchies. Some find it useful to partition the workload for these benchmarks, putting the control tasks on a general-purpose processor and the computationally intensive tasks on a hardware accelerator, coprocessor, or DSP engine.

DENBench Suite

Derived from industry sources following a mandate from members that they be written in relatively strict ANSI C, developers can use the DENBench suite for most processing platforms.

The suite consists of 14 benchmarks. Each benchmark must run through five to seven different data sets, yielding a total of 69 tests. EEMBC provides benchmark scores for the individual benchmarks and divides the suite into four minisuites, each with its own mathematically derived benchmark scores. The DENmark suite provides a geometric mean of all four minisuites.

MPEG EncodeMark—Measures motion-video encoding and has a geometric mean of 10 tests × 1,000; consists of two benchmarks:

- MPEG-2 video encoding
- MPEG-4 video encoding

MPEG DecodeMark—Measures motion-video and digital-audio decoding and has a geometric mean of 15 tests × 1,000; consists of the following benchmarks:

- MPEG-2 video decoding
- MPEG-4 video decoding
- MPEG-2 Layer 3 MP3 player audio decoding

CryptoMark—Measures data encryption and decryption and has a geometric mean of 4 tests × 10; consists of the following benchmarks:

- AES, the Advanced Encryption Standard;
- DES, the Data Encryption Standard;
- RSA, Rivest-Shamir-Adleman public-key cryptography; and
- Huffman decoding for data decompression

ImageMark—Measures still-image compression, decompression, and color-space conversion and has a geometric mean of 35 tests × 10; consists of the following benchmarks:

- JPEG Compress—performs still-image data compression using the JPEG standard;
- JPEG Decomp—performs still-image data decompression using the JPEG standard;
- RGB to CMYK—converts red-green-blue color space to cyan-magenta-yellow-black color space;
- RGB to YIQ—converts red-green-blue color space to a luminance-chrominance color space; and
- RGB to HPG—converts red-green-blue color space to Hewlett-Packard graphics

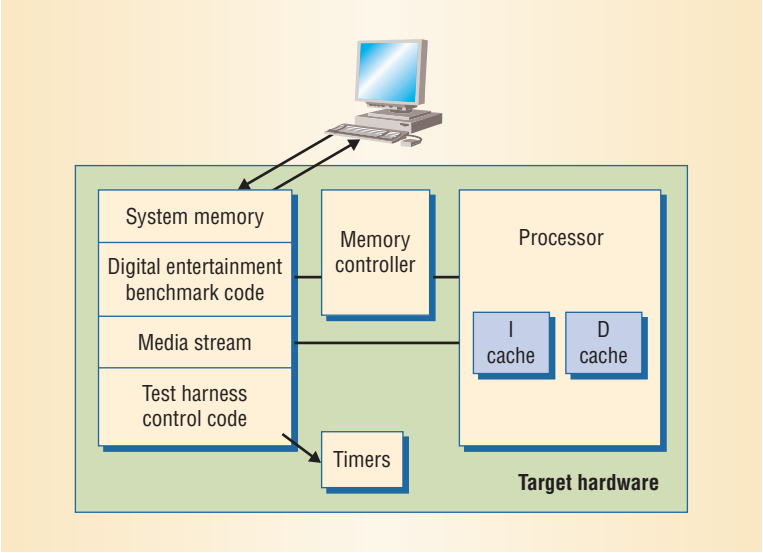


Figure 1. A common framework, or test harness, implements the EEMBC benchmarks. A host computer can use the framework to communicate to a device under test through most types of interfaces, including serial, parallel, and PCI ports.

DEFINING THE BENCHMARKS

As expected with a consortium, defining a benchmark involves a lengthy and arduous process. Indeed, the process for these newly released media benchmarks began nearly three years ago. At the earliest stages of development, selecting a representative set of applications to sufficiently test the processors and systems under consideration provided the first challenge. The next challenge involved determining how to make these benchmarks portable enough to run on the wide variety of processors and configurations that constitute the targeted digital consumer applications.

Many popular benchmarks perform a fixed workload. Throughput benchmarks, on the other hand, have no concept of finishing a fixed amount of work. Developers use throughput benchmarks to measure the rate at which work gets done. EEMBC's approach has always been based on a fixed workload.

Using the MPEG-x benchmark as an example, a fixed workload approach would process a video with a specific number of frames, measuring how long it took to process the entire video. Alternatively, running the benchmark for a fixed amount of time would measure the number of frames processed.

Although EEMBC implements the former approach, a potential issue arises because fixed-work benchmarks become outdated when computing capability or cache and memory capacity increase—which occurred with some of EEMBC's first-generation benchmarks. The tricky balance involves making the input data set large enough to thwart the most robust processors and systems but also small enough for use with lower-end platforms or to run on simulated environments.

When developing video codec benchmarks, the encoder and decoder portions must be measured separately. These two elements stress different parts of the CPU platform and are used in different numbers. For example, decoders are more common than encoders, and most devices can only decode.

CREATING PORTABLE BENCHMARKS

Creating portable benchmarks, whether using MediaBench or the DENBench suite, requires using a high-level language to stress compilation technology. Derived from industry sources, all benchmarks are written in relatively strict ANSI C. Theoretically, researchers can use ANSI C to compile benchmarks for most processing platforms, although hand coding will be required to make benchmark optimizations that run on an offload engine.

The DENBench suite runs natively, directly on the processor hardware and without an operating system. Although this represents a deviation from most real-world implementations, it supports portability because it eliminates any operating system dependencies or application-programming interface issues.

EEMBC implements all of its benchmarks using a common framework, or test harness, to ensure that everyone runs the benchmarks similarly. This test harness, which runs on a host computer, communicates with the target platform and extracts a hardware adaptation layer for easy porting, as Figure 1 shows.

A truly portable benchmark ensures that users run it consistently, compile the same code, and process the same workload. A disadvantage of such benchmarks is that few, if any, users would ever implement this type of code in a real system, especially a performance-, power-, and memory-constrained system.

To accommodate real-world implementations, EEMBC deviates from other benchmark approaches by supporting both out-of-the-box portable code and optimized implementations. Researchers can modify the benchmark code of an optimized implementation to take advantage of hardware accelerators or coprocessors or special instructions, or even to utilize intrinsics that will extend the compiler's functionality. Out-of-the-box benchmarks cannot take advantage of a multiprocessing or multithreading system's resources, however, which is especially disadvantageous with media benchmarks that support a significant amount of parallelism.

DERIVING BENCHMARKS FOR MULTIPROCESSING

Developers refer to the two fundamental multiprocessing methods as symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP). Also known as hetero- geneous multiprocessing, with AMP the system designer can use the processor best suited for a specific task. For example, the majority of mobile phones use the Texas Instruments AMP, the open multimedia appli- cations platform. OMAP consists of an ARM core and a DSP. The platform uses the ARM processor to run the application code and user interface, and it uses the DSP for modem functions and accelerating multimedia algorithms such as MPEG-4 decode.

Theoretically, developers can rewrite the DENBench code to accommodate either SMP or AMP, but this requires a relatively manual process.

Although realizing any of these multiprocessing methods in hardware is relatively straightforward, the true challenge is to make multiprocessing transparent to the system designer or benchmark porter. Ultimately, system designers will be able to implement multiprocessing applications with minimal effort, which motivates developers to place the partitioning burden on the operating system, compiler, and other software tool vendors.

A simple, albeit ineffective, solution to benchmarking multiprocessor systems would be to measure the total throughput available from running multiple instances of either the same or different applications, eliminating any interapplication dependencies. However, creating realistic usage scenarios for running multiple independent applications requires extra effort. Specifically, any benchmark should run different applications to stress the platform's ability to support the multiple cache contexts associated with multiple applications, as opposed to highlighting the ability to cache a single application across multiple processors.

Developers can derive an efficient multiprocessing benchmark by parallelizing a single task to scale across multiple instruction contexts. This type of benchmark must stress the system in terms of fine-grained synchronization access to shared resources. But there's more to it than that. A specific SMP benchmark suite must be on hand, rather than just an SMP optimization of existing suites.

Parallelism has three potential characterizations in this context:

- *task decomposition* parallelizes a single algorithm to share its workload over multiple processors;
- *multiple algorithms* look more at how the bigger system, including the OS, handles the workload from multiple concurrent algorithms; and
- *multiple streams* look at the bigger system, but concentrate more on data throughput and how a system can handle multiple data channels.

The true challenge is to make multiprocessing transparent to the system designer or benchmark porter.

If we look solely at task decomposition, we could consider SMP simply as an optimization of an existing suite's algorithms. For example, we could parallelize each of the kernels in the DENBench to use multiple processors. But for the other two forms of parallelism, this does not really fit.

Existing SMP benchmarks such as SPECint-Rate (www.at.nwu.edu/rtg/sp/hardware_performance.ssi) consider some aspect of multiple streams by looking at the throughput of an MP system through a simple duplication of a smaller test run in multiple instances simultaneously. This does not meet EEMBC's real-world criteria because it fails to consider the decomposition of multiple streams or interstream synchronization as part of the test result.

Instead, in the key SMP markets, a device must perform more than one simultaneous function. This is evident today, for example, when someone uses a mobile phone to place a video call. In this case, the system processes encode-decode algorithms along with the user interface and, possibly, networking protocols. The SMP benchmark suite must evaluate the stress points on the hardware and OS, while supporting these multiple code and data working sets.

DEFINING CONFORMANCE AND QUALITY

Developers can optimize the EEMBC benchmarks to take better advantage of each platform's feature set. Beyond performance testing, however, we must also consider the matter of defining conformance and quality testing if we focus on anything other than a standard encoding algorithm. Some would favor avoiding anything except a precisely defined encoding algorithm, while most would like to implement the encoder solution that best fits their architecture.

An encoder's output cannot be verified easily because, for example, an MPEG-2-compliant output with varying degrees of quality can be created in several ways. The output can require drastically different CPU times depending on the motion search algorithm, range, and so on.

Other than heuristics such as the peak signal-to-noise ratio and perceptual quality adaptation, there is no standard for checking the output's quality. Industry standards define decoders and the compliance rules for their outputs more tightly. Developers can also apply the peak signal-to-noise ratio test to these outputs.

Performance and quality provide good starting points for evaluating the capabilities of a digital entertainment system, but energy consumption is an equally important metric.

Currently, EEMBC is in the final stages of developing a standardized method for simultaneously measuring power and performance on processor-based systems. Many questions must be answered in developing a standard for measuring a processor's power and energy consumption. For example, at the most basic level, every vendor has a slightly different definition of what constitutes a processor: Is it comprised of the core alone or does it include cache and peripherals? Further, power consumption can vary significantly with different cache configurations, system bus loading, and especially the type of application code running. Thus, it is important to characterize the target system's power consumption rather than relying on device specifications.

Another challenge arises because every vendor runs its processors using a different workload, and, typically, these workloads are artificial constructs created to simulate typical and worst-case scenarios. The EEMBC benchmarks guarantee a common set of workloads, but there is also enough variety in the types of benchmarks to demonstrate the power a processor's different architectural features consume.

This standardization will let processor vendors and system designers test out the exquisite power-saving features. This additional metric will help classify processors for the appropriate target markets, such as portable versus line-powered, and will demonstrate that having the fastest processor is not necessarily best in the digital entertainment market. ■

References

1. L. Eeckhout et al., "Control Flow Modeling in Statistical Simulation for Accurate and Efficient Processor Design Studies," *Proc. 31st Ann. Int'l Symp. Computer Architecture*, IEEE Press, 2004, p. 350.
2. R.H. Bell Jr. and L.K. John, "Experiments in Automatic Benchmark Synthesis," Tech. Report TR-040817-01, Laboratory for Computer Architecture, Univ. of Texas at Austin, 2004.
3. C. Lee, M. Potkonjak, and W.H.M. Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communication Systems," *Proc. 30th Int'l Symp. Microarchitecture*, ACM Press, 1997, pp. 330-335.

Markus Levy *is founder and president of the Embedded Microprocessor Benchmark Consortium. Contact him at markus@eembc.org.*

CALLS FOR IEEE CS PUBLICATIONS

IEEE Annals of the History of Computing is planning a **January-March 2007** special issue on the computer communications services and technologies that existed before the development of the Internet. *Annals* seeks papers on bulletin boards, dial-up servers, and communications packages on time-share systems; store and forward networks, pre-OSI communications protocols, Bitnet, Csnet, Telenet, Prodigy, Fidonet, CompuServe, and other networks; early use of Listserv and similar protocols; and any technology providing a service that has since been replaced by software running on the Internet.

Interested authors should contact *Annals* Editor in Chief David Alan Grier at grier@gwu.edu. Abstracts are due by **1 September**. To view the complete call for papers, visit www.computer.org/portal/pages/annals/content/cfp07.html.

IEEE Software magazine plans a **July/August 2006** special issue on software verification and validation techniques. *Software* seeks papers on topics that include the automation of software testing, experiences in testing process improvement, testing metrics, and best practices for testing in specific domains.

Software focuses on providing its readers with practical and proven solutions to real-life challenges.

Complete author instructions are available at www.computer.org/software/author.htm#Submission. Submissions are due by 1 November. View the complete call for papers at www.computer.org/software/edcal.htm.

Submission Instructions

The Call and Calendar section lists conferences, symposia, and workshops that the IEEE Computer Society sponsors or cooperates in presenting. Complete instructions for submitting conference or call listings are available at www.computer.org/conferences/submission.htm.

A more complete listing of upcoming computer-related confeences is available at www.computer.org/conferences/.

OTHER CALLS

HPCA-12, 12th IEEE Int'l Symp. on High-Performance Computer Architecture, 11-15 Feb. 2006, Austin, Texas. Abstract due **11 July;** papers due **18 July.** www.cse.psu.edu/conf/hpca/

IEEE VR 2006, IEEE Virtual Reality Conf., 25-29 Mar. 2006, Alexandria, Va. Submissions due **4 September.** www.vr2006.org/cfp.htm#paper

DSN 2006, Int'l Conf. on Dependable Systems & Networks, 25-28 June 2006, Philadelphia. Abstracts due **18 November.** www.dsn2006.org/

ICIS 2006, 5th Int'l Conf. on Computer & Information Science, 12-14 July 2006, Honolulu. www.acisinternational.org/. Full paper submissions due **1 February-1 April 2006.**

CALENDAR

AUGUST 2005

2-4 Aug: ICCNMC 2005, Int'l Conf. on Computer Networks & Mobile Computing, Zhangjiajie, China. www.iccnmc.org/

4-5 Aug: MTDT 2005, IEEE Int'l Workshop on Memory Technology, Design, & Testing, Taipei, Taiwan. http://ats04.ee.nthu.edu.tw/~mtdt/

8-10 Aug: ICCI 2005, 4th IEEE Int'l Conf. on Cognitive Informatics, Irvine, Calif. www.enel.ucalgary.ca/ICCI2005/

8-11 Aug: CSB 2005, IEEE Computational Systems Bioinformatics Conf., Palo Alto, Calif. http://conferences.computer.org/bioinformatics/

14-16 Aug: Hot Chips 17, Symp. on High-Performance Chips, Palo Alto, Calif. www.hotchips.org/

17-19 Aug: Hot Interconnects, 13th IEEE Symp. on High-Performance Interconnects, Palo Alto, Calif. www.hoti.org/

17-19 Aug: RTCSA 2005, 11th IEEE Int'l Conf. on Embedded & Real-Time Computing Systems & Applications, Hong Kong. www.comp.hkbu.edu.hk/~rtcsa2005/

29 Aug.-2 Sept: RE 2005, 13th IEEE Int'l Requirements Eng. Conf., Paris. http://crinfo.univ-paris1.fr/RE05/

SEPTEMBER 2005

7-9 Sept: SEFM 2005, 3rd IEEE Int'l Conf. on Software Eng. & Formal Methods, Koblenz, Germany. http://sefm2005.uni-koblenz.de/

12-14 Sept: IWCW 2005, 10th Int'l Workshop on Web Content Caching & Distribution, Sophia Antipolis, France. http://2005.iwcw.org/

15-16 Sept: AVSS 2005, Conf. on Advanced Video & Signal-Based Surveillance, Como, Italy. www-dsp.elet.polimi.it/avss2005/

17-21 Sept: PACT 2005, 14th Int'l Conf. on Parallel Architectures & Compilation Techniques, St. Louis. www.pactconf.org/pact05/

18-21 Sept: CDVE 2005, 2nd Int'l Conf. on Cooperative Design, Visualization, & Eng., Palma de Mallorca, Spain. www.cdve.org/

19-21 Sept: CODES + ISSS 2005, Int'l Conf. on Hardware/Software Codesign & System Synthesis, Jersey City, N.J. www.codes-isss.org/

19-22 Sept: Metrics 2005, 11th IEEE Int'l Software Metrics Symp., Como, Italy. http://metrics2005.di.uniba.it/

19-22 Sept: WI-IAT 2005, IEEE/WIC/ACM Int'l Joint Conf. on Web Intelligence & Intelligent Agent Technology, Compiegne, France. www.comp.hkbu.edu.hk/WI05/

19-23 Sept: EDOC 2005, 9th Int'l Conf. on Enterprise Computing, Enschede, Netherlands. http://edoc2005.ctit.utwente.nl/

20-22 Sept: WRAC 2005, 2nd IEEE/NASA/IBM Workshop on Radical Agent Concepts, Greenbelt, Md. http://aaaprod.gsfc.nasa.gov/WRAC/home.cfm

21-24 Sept: VL/HCC 2005, IEEE Symp. on Visual Languages & Human-Centric Computing, Dallas. http://viscomp.utdallas.edu/vlhcc05/

23-24 Sept: ISoLA 2005, Workshop on Leveraging Applications of Formal Methods, Verification, & Validation, Columbia, Md. www.technik.uni-dortmund.de/tasm/isola2005/html/index.html

25-30 Sept: ICSM 2005, 21st IEEE Int'l Conf. on Software Maintenance, Budapest. www.inf.u-szeged.hu/icsm2005/

25 Sept: Vissoft 2005, 3rd Int'l Workshop on Visualizing Software for Understanding & Analysis (with ICSM), Budapest. www.sdml.info/vissoft05/

26-29 Sept: MASCOTS 2005, Int'l Symp. on Modeling, Analysis, & Simulation of Computer & Telecomm. Systems, Atlanta. www.mascots-conference.org/

27-30 Sept: Cluster 2005, IEEE Int'l Conf. on Cluster Computing, Boston. www.cluster2005.org/

30 Sept.-1 Oct: SCAM 2005, 5th IEEE Int'l Workshop on Source Code Analysis & Manipulation (with ICSM), Budapest. www.dcs.kcl.ac.uk/staff/mark/scam2005/

OCTOBER 2005

2-5 Oct: ICCD 2005, Int'l Conf. on Computer Design, San Jose, Calif. www.iccd-conference.org/

2-7 Oct: MoDELS 2005, 8th IEEE/ACM Int'l Conf. on Model-Driven Eng. Languages & Systems (formerly UML), Montego Bay, Jamaica. www.umlconference.org/

3-5 Oct: DFT 2005, 20th IEEE Int'l Symp. on Defect & Fault Tolerance in VLSI Systems, Monterey, Calif. www3.deis.unibo.it/dft2005/

3-7 Oct: BroadNets 2005, 2nd IEEE Int'l Conf. on Broadband Networks, Boston. www.broadnets.org/

5-8 Oct: ISMAR 2005, 4th IEEE & ACM Int'l Symp. on Mixed & Augmented Reality, Vienna. www.ismar05.org/

7-8 Oct: GridNets 2005, 2nd Int'l Workshop on Networks for Grid Applications (with BroadNets 2005), Boston. www.gridnets.org/

6-8 Oct: WWC 2005, IEEE Int'l Symp. on Workload Characterization, Austin, Texas. www.iiswc.org/iiswc2005/

10-12 Oct: DS-RT 2005, 9th IEEE Int'l Symp. on Distributed Simulation & Real-Time Applications, Montreal. www.cs.unibo.it/ds-rt2005/

12-14 Oct: HASE 2005, 9th IEEE Int'l Symp. on High-Assurance Systems Eng., Heidelberg, Germany. http://hase.informatik.tu-darmstadt.de/

15-21 Oct: ICCV 2005, 10th IEEE Int'l Conf. on Computer Vision, Beijing. www.research.microsoft.com/iccv2005/

17-19 Oct: BIBE 2005, IEEE 5th Symp. on Bioinformatics & Bioeng., Minneapolis. www.bibe05.org/

18-20 Oct: ICEBE 2005, IEEE Int'l Conf. on e-Business Eng., Beijing. www.cs.hku.hk/icebe2005/

18-21 Oct: ISWC 2005, 9th Int'l Symp. on Wearable Computers, Osaka, Japan. www.cc.gatech.edu/ccg/iswc05/

19-21 Oct: AIPR 2005, 34th IEEE Applied Imagery Pattern Recognition Workshop, Washington, D.C. www.aipr-workshop.org/

19-22 Oct: FIE 2005, Frontiers in Education Conf., Indianapolis, Ind. http://fie.engrng.pitt.edu/fie2005/

19-22 Oct: Tapia 2005, Richard Tapia Celebration of Diversity in Computing Conf., Albuquerque, N.M. www.ncsa.uiuc.edu/Conferences/Tapia2005/

23-25 Oct: FOCS 2005, 46th Ann. IEEE Symp. on Foundations of Computer Science, Pittsburgh. www.cs.cornell.edu/Research/focs05/

23-28 Oct: IEEE Visualization 2005, Minneapolis. http://vis.computer.org/vis2005/

26-28 Oct: ANCS 2005, Symp. on Architectures for Networking & Comm. Systems, Princeton, N.J. www.cesr.ncsu.edu/ancs/

26-28 Oct: SRDS 2005, 24th Int'l Symp. on Reliable Distributed Systems, Orlando, Fla. http://srds05.csee.wvu.edu/

NOVEMBER 2005

2-4 Nov: BTW 2005, IEEE 4th Int'l Workshop on Board Test, Fort Collins, Colo. www.molesystems.com/BTW05/

2-4 Nov: MTV 2005, 6th Int'l Workshop on Microprocessor Test & Verification, Austin, Texas. http://mtv.ece.ucsb.edu/MTV/

6-9 Nov: ICNP 2005, 13th IEEE Int'l Conf. on Network Protocols, Boston. http://csr.bu.edu/icnp2005/

6-10 Nov: ICCAD 2005, IEEE/ACM Int'l Conf. on Computer-Aided Design, San Jose, Calif. www.iccad.com/

7-10 Nov: MASS 2005, 2nd IEEE Int'l Conf. on Mobile Ad Hoc & Sensor Systems, Washington, D.C. www.mass05.wpi.edu/

7-11 Nov: ASE 2005, 20th IEEE/ACM Int'l Conf. on Automated Software Eng., Long Beach, Calif. www.ase-conference.org/

8-10 Nov: ITC 2005, Int'l Test Conf., Austin, Texas. www.itctestweek.org/

8-11 Nov: ISSRE 2005, 16th IEEE Int'l Symp. on Software Reliability Eng., Chicago. www.issre.org/

10-11 Nov: SDD 2005, 2nd IEEE Int'l Workshop on Silicon Debug & Diagnosis, Austin, Texas. http://evia.ucsd.edu/conferences/sdd/05/

12-16 Nov: Micro 2005, 38th ACM/IEEE Int'l Symp. on Microarchitecture, Barcelona, Spain. http://pcsostres.ac.upc.edu/micro38/

12-18 Nov: SC 2005, Seattle. http://sc05.supercomputing.org/

14-16 Nov: ICTAI 2005, 17th Int'l Conf. on Tools with AI, Hong Kong. http://ictai05.ust.hk/

15-17 Nov: LCN 2005, 30th IEEE Conf. on Local Computer Networks, Sydney, Australia. www.ieeelcn.org/

17-18 Nov: ISESE 2005, ACM-IEEE 4th Int'l Symp. on Empirical Software Eng., Noosa Heads, Australia. http://attend.it.uts.edu.au/isese2005/

26-30 Nov: ICDM 2005, 5th IEEE Int'l Conf. on Data Mining, New Orleans. www.cacs.louisiana.edu/~icdm05/

28-30 Nov: WMTE 2005, 3rd IEEE Int'l Workshop on Wireless & Mobile Technologies in Education, Tokushima, Japan. http://lttf.ieee.org/wmte2005/

DECEMBER 2005

4-7 Dec: WSC 2005, Winter Simulation Conf., Orlando, Fla. www.wintersim.org/

5-8 Dec: RTSS 2005, 26th IEEE Real-Time Systems Symp., Miami Beach, Fla. www.rtss.org/

12-14 Dec: ISM 2005, Int'l Symp. on Multimedia, Irvine, Calif. http://ism2005.eecs.uci.edu/

18-21 Dec: ISSPIT 2005, IEEE Symp. on Signal Processing & Information Technology, Athens. www.isspit.org/

18-21 Dec: ATS 2005, IEEE 14th Asian Test Symp., Kolkata, India. www.iitkgp.ac.in/ats05/

18-21 Dec: HIPC 2005, 12th Ann. IEEE Int'l Conf. on High-Performance Computing, Goa, India. www.hipc.org/hipc2005/index.html

19-21 Dec: CollaborateCom 2005, Int'l Conf. on Collaborative Computing: Networking, Applications, & Worksharing, Cape Cod, Mass. www.collaboratecom.org/

JANUARY 2006

5-7 Jan: Tabletop 2006, First IEEE Int'l Workshop on Horizontal Interactive Human-Computer Systems, Adelaide, Australia. www.tinmith.net/tabletop2006/

8-11 Jan: Key West 2006, IEEE Key West Computer Elements Workshop, Key West, Fla. www.unf.edu/ccec/ieee/IEEE-2006-KeyWest-Call.pdf

17-19 Jan: Delta 2006, Int'l Workshop on Electronic Design, Test, & Applications, Kuala Lumpur, Malaysia. www.monash.edu.my/events/delta2006/

23-27 Jan: Saint 2006, Symp. on Applications & the Internet, Phoenix, Ariz. http://infonet.cse.kyutech.ac.jp/conf/saint06/

FEBRUARY 2006

11-15 Feb: HPCA-12, 12th IEEE Int'l Symp. on High-Performance Computer Architecture, Austin, Texas. www.cse.psu.edu/conf/hpca/

MARCH 2006

13-17 Mar: PERCOM 2006, 4th Ann. IEEE Int'l Conf. on Pervasive Computing & Comm., Pisa, Italy. http://cnd.iit.cnr.it/percom2006/

2005 Frontiers in Education Conference

The 2005 Frontiers in Education Conference continues a long tradition of promoting the widespread dissemination of innovations in computer science, engineering, and technology education. Conference organizers have invited papers on topics that include active learning, entrepreneurship programs, K-12 initiatives and partnerships, and creative ways to teach and assess ethics.

FIE 2005 takes place from **19-22 October** in Indianapolis, Indiana. A series of workshops is scheduled for **19 October**, immediately preceding FIE 2005.

For further conference details, including registration information as it becomes available, visit http://fie.engrng.pitt.edu/fie2005/.

Student Design Teams Compete for $20,000 Prize

Sixth annual IEEE Computer Society International Design Competition welcomes teams from all areas of the world, including Romania, the United Arab Emirates, China, and Colombia.

Teams of undergraduate engineering students from colleges around the world meet in Washington, DC, each year to compete in a computer design event that caps off two semesters of extracurricular effort. To participate in the IEEE Computer Society International Design Competition, students design and implement computer-based solutions to real-world problems.

The 2005 competition focuses on the theme "Going Beyond the Boundaries." Accordingly, organizers have expanded this year's competition to include one contestant per team who is an undergraduate in a field outside computing.

An international panel of judges has selected 10 teams from nearly 200 original contenders to face off for two intensive days of competition. In May, judges reviewed project reports that contained specifications for the systems, including engineering considerations and implementation plans. A number of the original teams failed to pass an interim report stage in February, were eliminated through intraschool competitions, or dropped out before submitting the required 20-page report. CSIDC rules allow only one team per school to advance to the final stages of the competition.

Entries on display at the World Finals include an animal tracking system, several solutions for blind or deaf people, and a flood prediction device. The "CSIDC 2005 Names Top Ten Finalist Teams" sidebar lists the CSIDC 2005 World Finals teams and project names.

Said Alan Clements, chair of the CSDIC Committee for five years and a professor at the University of Teesside in England, "The success of earlier CSIDC competitions has been due to the wide range of projects that have been submitted. CSIDC 2005 expands this theme by encouraging teams to include members who are not computer science majors. This expansion and interdisciplinary approach has not only provided an exciting set of finalists but has also challenged the teams to work together despite the increased diversity. This is critical because teams are judged not only on technical merits but also on teamwork."

Clements also made note of the increased expectations of CSIDC judges, "Year by year, I have seen the quality of the reports submitted improve. Teams are sending in final reports that are remarkable for the quality of their presentation skills. However, it is now necessary to raise the standards by which reports are evaluated; in fact, we now expect students to appreciate the legal implications of their projects.

CSIDC 2005 Names Top Ten Finalist Teams

These 10 teams have been selected by an international panel of judges to present finished versions of their systems at the 2005 IEEE Computer Society International Design Competition World Finals live event. CSIDC organizers require that undergraduate teams work cooperatively in creating their entries. This year's competition theme is "Going Beyond the Boundaries."

- American University of Sharjah (United Arab Emirates), "ABBAS: Automobile Black Box for Accident Simulation"
- Beijing University of Technology (China), "Sporting Personal Assistant"
- Iowa State University (USA), "Lost in the Information World"
- North Carolina State University (USA), "NEAT: Networks for Endangered Animal Tracking"
- Politehnica University of Bucharest (Romania), "NOMAD Positioning System"
- Poznan University of Technology (Poland), "Read IT: A Portable Text Reading System for Blind People"
- Shanghai JiaoTong University (China), "Currahee NetMeeting System"
- Sir Syed University of Engineering Technology (Pakistan), "Boltay Haath—Pakistan Sign Language Recognition"
- SSN College of Engineering & Panimalar Engineering College (India), "VISION: Engineering Solutions for the Visually Challenged"
- Pontifical University of Bolivariana (Colombia), "ISPI: Intelligent System to Predict Inundations"

For example, judges might ask, 'Who does this data belong to?' or 'What happens if your system fails?'"

Success at CSIDC pays handsomely. Prizes range from $20,000 for first place to $10,000 for third place, plus honorable mention awards of $2,500. The competition also includes two $3,000 special prizes: the Microsoft Award for Software Engineering and the Microsoft Multimedia Award.

Primary financial support for CSIDC 2005 is provided by Microsoft, which has provided $1 million in funding for CSIDC until 2006. Zurich-based ABB Group has provided additional financial support. ■

2005 Ombudsman Serves as Member Advocate

In an effort to provide a single point of contact for member service issues, the IEEE Computer Society Board of Governors has established the position of ombudsman. Each year, the Society's Board of Governors appoints a volunteer to this post. The Computer Society ombudsman is charged with reviewing and responding to member complaints regarding service and subscription issues.

The IEEE Computer Society Policies and Procedures Manual contains a detailed description of the ombudsman's role, reproduced here, in part:

Section 26: Membership Committee

26.1 Ombudsman

26.1.1 Background

The idea of an ombudsman was raised at an Executive Committee meeting after hearing of a number of complaints from members such as not receiving the journals or magazines that they had ordered, or not having their membership/dues status acknowledged.

26.1.3 Duties and Functions

26.1.3.1 A copy of all Computer Society-related complaints received by the IEEE or Computer Society should be sent to the ombudsman. A standard form could be generated which indicates: the name/address of the member, the nature of the complaint, and the action instigated to rectify the problem. The ombudsman would not normally be involved with normal non-fulfillment complaints, except to receive a copy of the form.

26.1.3.4 Members are invited to write directly to the ombudsman if they have reason to believe their original complaint has not received the attention it deserves. The ombudsman is responsible for:

a. immediately acknowledging receipt of the complaint.
b. investigating the nature of the complaint and investigating whatever action is necessary to rectify the problem.
c. responding to the member with details of the actions taken and inviting he/she to correspond further if either this action does not solve the problem or if he/she still remains dissatisfied.

26.1.3.5 The ombudsman should report to the Membership Committee but has direct access to the Board of Governors with respect to any unusual or otherwise important complaints which are not readily rectified, except that this shall not apply to those portions of Computer Society operations where procedures for appeal already exist.

2005 Ombudsman Fiorenza Albert-Howard

Serving as ombudsman in 2005 is Fiorenza Albert-Howard of Capilano College in North Vancouver, Canada. She praised Computer Society staffers for handling complaints efficiently, while suggesting an expanded role for the ombudsman.

Said Albert-Howard, "The majority of our members are consulting the ombudsman only to report wrong addresses or ruined issues of the magazines to which they are subscribing. While this is an important part of the ombudsman's responsibilities, the staff of the Society handles the resolution of these issues very efficiently. The ombudsman also serves as a liaison between members, volunteers, and staff. Members should consider the ombudsman to be an accessible and approachable contact within the Society when they need help with an unresolved complaint."

Highlighting her personal commitment to the position of ombudsman, Albert-Howard continued, "I'm also available to field questions about the Society's bylaws, procedures, or rules and regulations and to provide other information about the Society that would be helpful to members."

Members can contact Albert-Howard at ombudsman@computer.org.■

IEEE Computer Society Election Update

Each year, all members of the IEEE Computer Society have an opportunity to vote for the officers who will plan new activities and direct the Society's operations in the coming year. Volunteer positions include leadership roles for the Publications, Educational Activities, and Electronic Products & Services Boards and membership on the IEEE Computer Society Board of Governors. The volunteers selected this year will serve under 2006 President Deborah Cooper, chosen last year as president-elect.

Society members can become candidates for office in one of two ways: by Nominations Committee recommendation or by petition. The Nominations Committee accepted member recommendations of candidates until April. At a June meeting, the current Board of Governors approved the following slate of candidates brought forward by the Nominations Committee.

PRESIDENT-ELECT AND VICE PRESIDENT CANDIDATES

The Board-approved candidates for 2006 president-elect/2007 president are Michael Williams and Yervant Zorian. The president supervises decisions that affect the Society's programs and operations and is a nonvoting member on most Society program boards and committees.

Candidates for first vice president are Rangachar Kasturi and Murali Varanasi. The second vice president candidates are Susan (Kathy) Land and Kathleen Swigger. After the elections, 2006 President Deborah Cooper will appoint the two elected vice presidents to oversee two Society boards. At her discretion, Cooper will select appointees to head up the Society's other governing boards.

BOARD OF GOVERNORS CANDIDATES

Members of the Board of Governors serve rotating three-year terms. The 14 candidates for 2006 to 2008 terms on the Board of Governors are Donald Bagert, Denis Baggi, Michael Blaha, Antonio Doria, Richard Eckhouse, James Isaak, Gary McGraw, James Moore, Sorel Reisman, Stephen Seidman, Robert Sloan, Pradip Srimani, Makoto Takizawa, and Stephanie White. The seven candidates who receive the most votes will assume seats on the Board starting in January 2006.

The IEEE Computer Society election window begins on 8 August, when paper ballots are mailed to all Society members, and ends on 4 October. All members will have the opportunity to vote via mail or online at www.computer.org/election/.

The paper ballots, the election area of the Society's Web page, and the September issue of *Computer* will provide biographical sketches and candidate position statements for each nominee. The biographical sketches will detail the candidates' IEEE Computer Society and other professional activities, current employment, professional experience and accomplishments, degrees and majors, awards, and honors.

We encourage all members to take part in electing the Computer Society's leaders. ■

Certification Testing Opportunities Resume on 1 September

Recognizing that, in a sea of product-based certifications and credentials, a standards-oriented proof of software engineering capabilities was needed, the IEEE Computer Society developed the Certified Software Development Professional program. Software engineers who earn the IEEE Computer Society CSDP credential can use it to verify their skills to current or potential employers.

Each year, the Computer Society offers two opportunities for members to take the CSDP exam: April through June, and September through November. Software engineers who hold a bachelor's degree and have a minimum of 9,000 hours of experience in the field are eligible to apply. In addition, candidates for certification must have had at least two years of software engineering experience within the four-year period prior to application.

Thomson Prometric administers the CSDP exam at test centers throughout the world, and new centers were added earlier this year (see "New CSDP Testing Sites Open in 2005," *Computer,* Mar. 2005, pp. 75-76). For IEEE or Computer Society members, 2005 CSDP exam fees total $450, including a $100 application fee and a $350 test administration fee. In 2004, the GI Bill Education Benefits Program approved CSDP credentialing fees as a reimbursable expense.

A CSDP study group online Yahoo forum, linked from the CSDP Web site, can help potential examinees to prepare. Resource materials and an online test preparation class are also available.

Applications for the 1 September through 30 November testing window must be postmarked by **1 September**. The application form is available online at www.computer.org/certification/bulletin.htm. For general information on the IEEE Computer Society CSDP program, visit www.computer.org/certification/.

Editor: Bob Ward, *Computer*, bnward@computer.org

JULY 2005

Advertiser / Product	Page Number
4DSP Inc.	83
Addison Wesley	**Cover 2**
AeroComm Inc.	83
Apress	84
Aventail Corp.	83
Cambridge University Press	84
CGO 2006	**Cover 3**
Chapman & Hall/CRC	84
Charles River Media	**15**
D.E. Shaw& Company	**81**
The Globus Alliance	83
ICDCS 2006	**19**
ICPP 2006	**27**
ICSE 2006	**9**
ICSM 2005	**5**
IEEE Computer Society Membership	**88-90**
Old Dominion University	**82**
Oxford University Press	84
Princeton University Press	84
Rochester Institute of Technology	**80**
Seapine Software, Inc.	**Cover 4**
TallyGenicom	83
Classified Advertising	***80-82***

Boldface denotes advertisements in this issue.

Computer
IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, California 90720-1314
USA
Phone: +1 714 821 8380
Fax: +1 714 821 4010
http://www.computer.org
advertising@computer.org

Advertising Sales Representatives

Mid Atlantic (product/recruitment)
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0161
Email: db.ieeemedia@ieee.org

New England (product)
Jody Estabrook
Phone: +1 978 244 0192
Fax: +1 978 244 0103
Email: je.ieeemedia@ieee.org

New England (recruitment)
Robert Zwick
Phone: +1 212 419 7765
Fax: +1 212 419 7570
Email: r.zwick@ieee.org

Northwest (product)
Peter D. Scott
Phone: +1 415 421 7950
Fax: +1 415 398 4156
Email: peterd@pscottassoc.com

Southeast (recruitment)
Thomas M. Flynn
Phone: +1 770 645 2944
Fax: +1 770 993 4423
Email: flynntom@mindspring.com

Midwest (product)
Dave Jones
Phone: +1 708 442 5633
Fax: +1 708 442 7620
Email: dj.ieeemedia@ieee.org

Will Hamilton
Phone: +1 269 381 2156
Fax: +1 269 381 2556
Email: wh.ieeemedia@ieee.org

Joe DiNardo
Phone: +1 440 248 2456
Fax: +1 440 248 2594
Email: jd.ieeemedia@ieee.org

Midwest/Southwest (recruitment)
Darcy Giovingo
Phone: +1 847 498 4520
Fax: +1 847 498 5911
Email: dg.ieeemedia@ieee.org

Southwest (product)
Josh Mayer
Phone: +1 972 423 5507
Fax: +1 972 423 6858
Email: jm.ieeemedia@ieee.org

Connecticut (product)
Stan Greenfield
Phone: +1 203 938 2418
Fax: +1 203 938 3211
Email: greenco@optonline.net

Southern CA (product)
Marshall Rubin
Phone: +1 818 888 2407
Fax: +1 818 888 4907
Email: mr.ieeemedia@ieee.org

Northwest/Southern CA (recruitment)
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Southeast (product)
Bill Holland
Phone: +1 770 435 6549
Fax: +1 770 435 0243
Email: hollandwfh@yahoo.com

Japan
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Europe (product/recruitment)
Hillary Turnbull
Phone: +44 (0) 1875 825700
Fax: +44 (0) 1875 825701
Email: impress@impressmedia.com

Advertising Personnel

Marion Delaney
IEEE Media, Advertising Director
Phone: +1 212 419 7766
Fax: +1 212 419 7589
Email: md.ieeemedia@ieee.org

Marian Anderson
Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: manderson@computer.org

Sandy Brown
IEEE Computer Society,
Business Development Manager
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sb.ieeemedia@ieee.org

DATABASE ADMINISTRATOR sought by computer consulting co. in Houston, TX to design database software applications. Requires Master's degree & exp. Respond by resume only to Mr. Scott Moore, #B/C-10, Enterprise Alliance Systems, Inc., 7457 Harwin Dr., Suite 252, Houston, TX 77036.

PROGRAMMER/ANALYST sought by InterNetwork Portal Co. in Houston, TX. To assist in developing web-based applications user/system documentation, and provide technical support. Requires degree in Computer Science & exp. Respond by resume only to Mr. Nasseem Rahman, OM, F/P-#10, Wirelessgalaxy.com, Inc., 7211 Regency Square, Suite 120, Houston, TX 77036.

CONSULTANT 1 (PeopleSoft CRM) (Unisys Corporation/Blue Bell, PA): manage, plan & coordinate full life cycle of highly customized & very large PeopleSoft CRM/Oracle DBA design & implementation projects. Reqs: master's degree or foreign equiv in comp sci + 3 yrs exp as Consultant for PeopleSoft/Oracle applications w/ lead or principle responsibility; position reqs 75% + travel throughout U.S. 40 hrs/wk 9-5; salary commensurate w/exp. Send resume to IEEE Computer Society, 10662 Los Vaqueros Circle, Box # COM7, Los Alamitos, CA 90720.

DATABASE ADMINISTRATOR sought to design & maintain database & accounting software packages. Requires degree & experience. Respond by resume only to Luis Moyano, IT Mgr., #J/Y-10, G & A Partners, 4801 Woodway, Ste 210, Houston, TX 77056.

DATA MODELER-Logical Data Modeler (DE) Transform project data requirements into project data models, facilitate JAD sessions, data analysis/database design technologies & tools in an operational/transactional environment. Min. 3 yrs exp. Fax resume to C. Marx 972-547-6429.

ENGINEER(S) Software Applications sought by company located in Studio City, CA. Bach degree or/equiv in Computer Science or, Software Eng. + 2 yrs. exp. specifically in "Magic" development. Mail resumes to: Ronen Canetti Wizmagic, LLC 4024 Radford Ave, Bldg 2. Room 201, Studio City, CA 91604.

CALTECH. Postdoctoral research position at Caltech's Center for Advanced Computing Research is open in the area of compiling for innovative, high performance computer architectures. Ph.D. in computer science, computer engineering or equivalent, and a strong background in languages and compilers for large-scale high performance scientific computation are required. See (www.cacr.caltech.edu/employment/cascade-postdoc.html). Resume to: Susan Powell, spowell@cacr.caltech.edu.

UNIVERSITY OF SASKATCHEWAN, Computer Engineering Assist. Professor. Full time tenure-track position in embedded systems. Ph.D. or equivalent is required and a vigorous research program is expected. Review of applications will begin September 15, 2005. http://www.engr.usask.ca/dept/ele/cmpe.

DATABASE ADMINISTRATOR required in Michigan FT 9:00am to 5:00pm. Bachelor degree of Science in Computer Science. Send resumes to: Att: Florentina J. Wood, R.N., Caring Nurses of Michigan, 29201 Telegraph Rd, Ste 505, Southfield, MI 48034.

PROGRAMMER ANALYST needed w/2 yrs exp to dvlp VBScripts, Kix Scripts & Wrappers for advanced Customs Actions for MSI packages. Implmt Applic Deployment products using Active Directory, Altiris Product Suites, CMS & SMS. Use SQL D/base, Wise Suite of Products incl Wise Package Studio, Wise for Win Installer running on Win 2000/2003 Servers & Win XP SP2 machines. Mail res to: Integration Specialist Inc, 355 Eisenhower Pkwy, Livingston, NJ 07039. Job Loc: Livingston, NJ.

DATABASE SYSTEM ADMINISTRATOR to design and administer database system for business application. Respond by resume only to Mr. R. Wu, #K/W-10, MODA Investment, Inc., 1500 N. Loop, Houston, TX 77009.

PROGRAMMER ANALYST FOR RELYCOM. Develop application software, setup definitions and processes, develop and review test cases for clients onsite. Req: BS - Info. Syst., experience. Must be willing to travel, accept Long/short trm. assignments. Must have exp. w/ REM tools, &, Doors. Apply to: S. Varanasi, Relycom Inc, 666 Plainsboro Rd, Ste 1171, Plainsboro, NJ 08536.

WEB DESIGNER: Create/dsgn layout for e-com/e-cont web/intra; underst broadband mkt & prodt; use comp softwr to gen image; knowl of graphic/web prgms; det size & arrangement of illus mater & copy; selt style/size; create chts; graph; illustra; artwk via comp; rev final layot; sugg imrovmts; confer w/clts; discus/det layot des; dev grapcs/layots for prod illustra; compy logo; website. 8 yrs wrk exp. Res: Nazir Madhani, Holly Brook Oil Corp., 1700 Douglas Rd., Miramar, FL 33025.

COMPUTER SYS. ANALYST LEV I wanted by educational company in Old Tappan, NJ. Must have Bach. Degree in Computer Science or Computer Engineering. Must speak, read, & write Korean. Apply to: Honors Review Learning Center, Old Tappan Center, 1 De Wolf Rd., Old Tappan, NJ 07675.

UNIVERSITY OF MICHIGAN-DEARBORN. The Electrical and Computer Engineering Department at the University of Michigan-Dearborn invites applications for a tenure track faculty position at the Assistant/Associate Professor Level. An earned Ph.D. in Electrical/Computer Engineering is required. The selected candidate will teach graduate and undergraduate courses in computer engineering and will actively pursue funded research in one or more areas of computer engineering. Selection of successful candidates is based on an assessment of teaching, research potential and prior experience. The University of Michigan-Dearborn is located in historic Dearborn, MI, the heart of the U. S. automotive industry. The University of Michigan-Dearborn is dedicated to the goal of building a culturally diverse and pluralistic faculty committed to teaching and working in a multicultural environment. Applications with a complete resume should be sent to: Chairman, ECE Department, University of Michigan-Dearborn, 4901 Evergreen Rd., Dearborn, MI 48128-1491. Applicants should clearly identify research interest and teaching experience. Applications will be accepted until the positions are filled. UM Dearborn is an equal opportunity employer and encourages applications from women and minorities.

DATABASE ADMINISTRATOR (Clifton, NJ) Coordinate d/base mgmt system, F/T Req. Bach/Comp Sci. Reply to: Jin-A Child Care Center, 77 Jay St., Clifton, NJ 07013.

SYSTEMS ANALYST - Farmingdale, NY. Software Applications Company seeks Systems Analyst to perform system requirements definition & detail system design REQ.: BS in Computer Science+2 yrs. experience developing PKI applications with RSA algorithm. M-F (9-5). Mail resume to: Juma Technology, 100 Broadhollow Rd., Farmingdale, NY 11735 or email fvinci@jumatechnology.com.

Boston, MA – SENIOR SOFTWARE ENGINEER sought to design, develop, and implement next generation products based on .Net technologies using advanced Software Architecture and Database Design skills; Develop appropriate mathematical solutions and algorithms based on business rules; Analyze, debug, and correct complex technical issues in a wide variety of technical disciplines; Design, develop, implement, and manage enterprise software systems using Microsoft SQL Server, SQL, VB, ASP, VB.NET, C#, ASP.NET, Visual Studio; Master's degree in information systems or related field required + 1 yr. exp.; M-F, 9-5. Mail resume to: Dir. Of HR, Risk Management Foundation of the Harvard Medical Institutions, Inc., 101 Main St., Cambridge, MA 02142.

SAP BUSINESS SYSTEMS ANALYST for transactions processing company. Requires minimally a Bachelor's degree in Computer Science or Management Information Systems and two years experience using SAP BW-SEM to design, develop and maintain SAP modules for corporate finance unit including performing business requirements analysis, designing & developing functional specifications and

data modeling for SAP implementation, coordinating system configuration and interface design, planning migration from legacy systems, executing QA testing processes and overseeing end user training. The position is located primarily in West Greenwich, R.I. with 10% domestic/international travel. Send resume to Human Resources, Attn: Denise Hempe, GTECH Corp., 55 Technology Way, West Greenwich, RI 02817.

SOFTWARE ENGINEER. Cary. Analyze, design, code, test and document proprietary software components using programming tools and applications including JAVA, EJB, XML, UNIX Websphere, Weblogic, IPlant and JBoss, Oracle, DB2, SQLServer and MySQL. Bach. degree in CS or related field with 2 yr exp. Send resume and salary requirement to Engineous Software, Inc. at jobs@engineous.com. Must reference "Software Engineer" on application.

SOFTWARE ENGINEERS/PROJECT MANAGERS - Enzo Solutions, LLC located in Weston, Florida seeks Software Engineers and Project Managers to apply the principles and techniques of computer science, engineering and mathematical analysis to analyze users' needs and design, create, and modify computer applications software and systems. Send resumes to info@enzosolutions.com.

OLD DOMINION UNIVERSITY

EXECUTIVE DIRECTOR
VIRGINIA MODELING, ANALYSIS & SIMULATION CENTER

Old Dominion University (ODU) invites applications for the position of Executive Director of a research center that emphasizes modeling, simulation, and visualization (MS&V) research, development and education. ODU's Virginia Modeling, Analysis & Simulation Center (VMASC) is one of the world's leading university research centers for computer modeling, simulation, and visualization. The mission of the Center is to conduct collaborative M&S research and development, provide expertise to industry and governmental agencies, and promote ODU, Hampton Roads and Virginia as a center of MS&V activities. The Center has over 50 research and administrative staff and works closely with faculty researchers from across the University. In 2004, the Center conducted approximately $10.5M in funded research. ODU offers master's and doctoral degrees in Modeling and Simulation supported by faculty from all six academic colleges and research faculty from VMASC. The program has an enrollment of approximately 55 master's and 45 doctoral students.

Old Dominion University is a state-assisted institution and one of only four Virginia schools in the Carnegie Foundation's "Doctoral/Research – Extensive" classification. It is located in Hampton Roads, the nation's center for the military application of MS&V. The region is home to the Joint War Fighting Center, the Joint Battle Center, the US Army's Training and Doctrine Command, the Military Transportation Management Command, the Armed Forces Staff College, the U. S. Navy's Commander Operational Test and Evaluation Force, the Naval Sea Systems Command, and the Space and Naval Warfare Center. In addition, Northrop Grumman, Jefferson Lab, and NASA – Langley Research Center are important users of M&S technology. The economic value of MS&V-related business activity in Hampton Roads is estimated to be over $500M. Leveraging the strength that has been brought about by VMASC, Virginia's Governor Mark R. Warner recently announced, and the General Assembly approved, a $1.45M state initiative to market and promote the region and establish a national Institute for Homeland Security and Crisis Management.

The responsibilities of the Executive Director include but are not limited to the following:

Providing leadership and vision for current and future research.
Developing major research initiatives that will result in extramural funding.
Building multidisciplinary teams and industrial partners/collaborations for funded research.
Directing and administering research activities and academic programs affiliated with VMASC.
Marketing VMASC's technical skills to industry and government agencies.

Candidates for the position of Executive Director must have a Ph.D., preferably in MS&V-related areas. Candidates must show evidence of: significant accomplishments in their area of expertise, excellence in acquiring and managing funded research, good interpersonal and interdisciplinary team-building skills, and demonstrated familiarity with government and industrial sponsors.

Salary is commensurate with experience and background. Tenure may be considered for those individuals whose credentials are appropriate for a tenured faculty position. A letter of application and a current resume with names, addresses, telephone numbers and email addresses for five references should be sent to: **Mohammad A. Karim, Vice President for Research, Old Dominion University, 2035 Hughes Hall, Norfolk, VA 23529. Tel: 757-683-3460; Email: mkarim@odu.edu**

Review of applications will begin immediately and continue until the position is filled.

Old Dominion University is an Affirmative Action/Equal Opportunity institution and requires compliance with the Immigration Reform and Control Act of 1986.

PROGRAMMER/ANALYST: To design & develop specialized enterprise applications for mortgage co. in Houston, TX. Requires B.S. in M.I.S. or Computer Science plus exp. Respond by resume only to: Robin T. Liggett, HR Mgr., # M/K-10, Aegis Mortgage Corp., 10049 N. Reiger Rd., Baton Rouge, LA 70809.

BUSINESS COMPUTER SUPPORT SPECIALIST sought by importers in Stafford, TX. Req'd degree in MIS. Respond by resume only to Mr. G. Kho, Pres., #J/L-10. Truemark Int'l Corp., 12503 Exchange Dr., Ste 506, Stafford, TX 77477.

COMPUTER SYSTEM DIRECTOR wanted by distributor of electronic items in Secaucus, NJ. Must have a Bachelor Degree in Computer Science or Computer Engineering. Must speak, read and write Korean. Apply to: Direct Plus Inc., 301 Penhorn Ave., Unit 5, Secaucus, NJ 07094.

ORACLE SYSTEM ADMINISTRATION ANALYST - Houston, Texas. Resp. for Oracle financial applications (security, concurrent manager and programs, alerts, workflow, adhoc report writing, troubleshooting interfaces), Oracle reporting tools, accounting modules and tools for database analysis, design and administration. Develop database links & snapshots between various Oracle and SQL Server databases. Create users, objects, grant and monitor user rights and privileges to ensure system security. Involved in system capacity planning and performance tuning databases. Responsible for physical and logical backup and recovery operations. Work with gas marketing, gas pipeline and natural gas liquid revenue accounting systems. Monitor and recommend improvements to Oracle Financial applications functionality, security and performance. Requires B.Sc. in Comp. Sci. or Electronics & 1 yr. exp. in job offered. Mail resumes to Enbridge Employee Services, Inc., Attn: Jennifer Williams, 1100 Louisiana Street, Suite 3300, Houston, Texas 77002. Use job code: HGLC.

Extreme Mobile Printing from TallyGenicom

TallyGenicom's new MTP4/MTP4R mobile thermal printers are built to survive extreme outdoor environments and rough handling, making them well suited for use in the direct-store delivery industry. They print up to 4 inches wide at 3 inches per second and feature an angled LCD display, easy drop-in paper loading, and wireless connectivity via USB, RS-232, Bluetooth, or IEEE 802.11b. Optional accessories include a magnetic stripe reader, a quick-release vehicle cradle, shoulder/hand straps, and a fast AC charger.

MTP4/MTP4R series printers start at $975 with battery included; www.tallygenicom.com.

Plug-and-Play RF Transceivers Get Ethernet Connectivity

AeroComm Inc.'s new ConnexNet Ethernet-enabled transceivers merge the company's radio frequency protocol with the Digi Connect ME embedded device server for global networking.

The product adds wireless network connectivity to any serial-based application, serving as a conduit between the user and multiple destination devices. The transceiver integrates a fully developed TCP/IP network stack and OS using various network protocols. Monitoring and controlling complete OEM networks is as easy as connecting to a local network or Internet portal from any location.

The ConnexNet CN4490-1000 (server/client) and CN4790-1000 (peer-to-peer) start at $199 each, while starter packs featuring one ConnexNet Ethernet device and one ConnexLink serial device cost $325 per system; www.aerocomm.com.

Globus Updates Toolkit for Grid Implementations

The Globus Alliance has released version 4.0 of the Globus Toolkit, an open source set of software services and libraries for building enterprise-level Grid systems and applications. GT4 complies with the latest WS-I Web services standards to facilitate interoperability between different environments, includes initial support for important authorization standards such as SAML and XACML to create a secure Web-services-enabled Grid infrastructure, and implements the emerging WS-RF and WS-N specifications.

Globus Toolkit 4.0 can be downloaded at www.globustoolkit.org.

IEEE-754 Compliant FFT Core from 4DSP

4DSP Inc. has released a floating-point fast Fourier transform core that is IEEE-754 compliant. Designed for new high-performance Xilinx and Altera field-programmable gate arrays, the FFT core performs transforms on complex data ranging from 256 to 1,000 points with external memory, if necessary, such as QDR SRAM, closely coupled to the FPGA's internal logic.

Based on the radix-32 butterfly architecture, the product lets users change the transform length without having to reconfigure the programmable device. This flexibility makes the core ideal for systems that change mission rapidly in application design or for complex algorithms like those used in high-precision spectral analysis, radar, and video processing.

Pricing for the FFT core starts at $23,000; www.4dsp.com/fft.htm.

Aventail SSL VPNs Add Smart Tunneling

Aventail Corp.'s newest VPN offering incorporates the company's Smart Tunneling technology, combining universal application access with cross-platform support, end-point control, and far greater security than traditional IPSec solutions. Smart SSL VPNs enable mobile workers to access corporate resources, file shares, and applications from all types of devices and include support for UDP, TCP, and IP protocols as well as back-connect applications such as those using VoIP.

Pricing starts at $6,995 for the EX-750, a full-featured, clientless SSL VPN appliance tailored to small to midsized enterprises; pricing for the Aventail EX-1500, a scalable, enterprise-class solution integrating high availability and load-balancing support, starts at $9,995; www.aventail.com.

TallyGenicom's MTP4/MTP4R mobile thermal printers are designed to withstand six-foot drops, rain, dust, and extreme temperatures.

Please send new product announcements to products@computer.org.

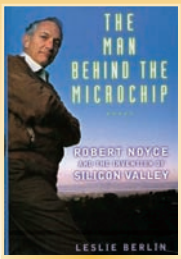
The Man Behind the Microchip: Robert Noyce and the Invention of Silicon Valley, Leslie Berlin. Hailed as the Thomas Edison *and* Henry Ford of Silicon Valley, Robert Noyce was an inventor, an entrepreneur, and a risk taker who piloted his own jets and skied mountains accessible only by helicopter. The author captures not only this colorful individual but also the vibrant interplay of technology, business, money, politics, and culture that shaped and still define Silicon Valley.

Cofounder of Fairchild Semiconductor and Intel, Noyce also coinvented the integrated circuit. Berlin paints a fascinating portrait of Noyce as an ambitious and intensely competitive multimillionaire who exuded a "just folks" charm, a Midwestern preacher's son who rejected organized religion but counseled his employees to "go off and do something wonderful," a man who never looked back and sometimes paid a price for it.

This vivid narrative also sheds light on Noyce's friends and associates, including some of the best-known managers, venture capitalists, and creative minds in Silicon Valley. Berlin draws upon interviews with dozens of key players in modern American business—including Andy Grove, Steve Jobs, Gordon Moore, and Warren Buffett.

Oxford University Press, www.oup.com; 0-19-516343-5; 440 pp.; $30.

A First Course in Scientific Computing: Symbolic, Graphic, and Numeric Modeling, Rubin H. Landau. This book offers a new approach to introductory scientific computing that aims to make students comfortable using computers. The author strives to provide readers with the computational tools and knowledge they will need throughout their college careers and into their professional careers and to show how all the pieces can work together. The text introduces the requisite mathematics and computer science through realistic problems—from energy use to building skyscrapers to



projectile motion with drag—then shows how each discipline uses its own language to describe the same concepts.

The book covers the basics of computation, numerical analysis, and programming from a computational science perspective. It uses the Maple problem-solving environment, moves on to the Java compiled language, and concludes with an introduction to LaTeX, replete with sample files.

Princeton University Press, http://pup.princeton.edu/; 0-691-09065-3; 472 pp.; $49.50.

Virtualization: From the Desktop to the Enterprise, Chris Wolf and Erick M. Halter. Creating a virtual network maximizes server use. This book demonstrates how to manage all aspects of virtualization across an enterprise, delving deeply into the technologies' interrelationships.

The authors cover both Microsoft and Linux environments, explore the many aspects of virtualization, including virtual machines, virtual file systems, virtual storage solutions, and clustering, and help readers understand which technologies might be right for their particular environment.

Apress; www.apress.com; 1-59059-495-9; 600 pp.; $59.99.

Mobile Web Services, Ariel Pashtan. Mobile Web services provide access to Web content anywhere and anytime. This book describes the key network elements, software components, and software protocols needed to realize these services, including the concept of user context and its potential to create personalized services.

The book examines mobile Web functions such as location representation and tracking, security schemes, content personalization, and XSLT processing for browser content generation. The author reviews the WAP and i-mode architectures, reviews the latest mobile phone features, and discusses key aspects of browser mark-up languages. The text covers the ontology concepts that enable the wireless semantic Web and offers a novel definition and categorization of mobile user context in RDF Schema.

Cambridge University Press; www.cambridge.org; 0-521-83049-4; 284 pp.; $60.

Distributed Sensor Networks, S. Sitharama Iyengar and Richard R. Brooks, eds. To create smart environments, researchers deploy thousands of sensors, each with a short-range wireless communications channel and capable of detecting ambient conditions such as temperature, movement, sound, light, or the presence of certain objects. With the emergence of high-speed networks and their increased computational capabilities, these distributed sensor networks have real-time applications in aerospace, automation, defense, medical imaging, robotics, and weather prediction.

This book offers the background theory and applications of this new technology. It provides essential coverage of wireless networks, signal processing, and self-organizing systems. Recurring themes include multidimensional data structures, reasoning with uncertainty, system dependability, and using metaheuristics.

Chapman & Hall/CRC; www.crcpress.com; 1-58488-383-9; 1144 pp.; $139.95.

Editor: Michael J. Lutz, Rochester Institute of Technology, Rochester, NY; mikelutz@mail.rit.edu. Send press releases and new books to *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; fax +1 714 821 4010; newbooks@computer.org.

Absolutely Positively on Time: What Would It Take?

Edward A. Lee, University of California, Berkeley

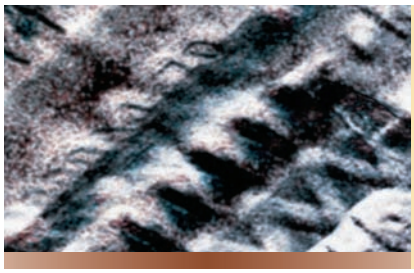
Despite considerable progress in software and hardware techniques, many recent computing advances do more harm than good when embedded computing systems absolutely must meet tight timing constraints.

For example, while synchronous digital logic delivers precise timing determinacy, advances in computer architecture and software have made it difficult or impossible to estimate or predict software's execution time. Moreover, networking techniques introduce variability and stochastic behavior, while operating systems rely on best-effort techniques. Worse, programming language semantics do not handle time well, so developers can only specify timing requirements indirectly.

Thus, achieving precise timeliness in a networked embedded system—an absolutely essential goal—will require sweeping changes.

CORE ABSTRACTION

Contemporary computer science has taught us that a Turing machine can specify everything that can be computed. Computation is accomplished by a terminating sequence of state transformations. A *computable function* provides a map from a bit sequence to a bit sequence. This core abstraction underlies the design of most computers, programming languages, and operating systems currently in use.



For embedded computing to realize its full potential, we must reinvent computer science.

Unfortunately, it does not fit embedded software well. If, however, "embedded software" is simply "software on small computers," then this abstraction fits *reasonably* well. In this view, embedded software differs from other software only in its resource limitations: small memory, small data word sizes, and relatively slow clocks. In this view, the *embedded software problem* is one of optimization.

Optimizing solutions emphasize efficiency: Engineers write software at a very low level in assembly code or C, avoid operating systems with a rich suite of services, and use specialized computer architectures such as programmable DSPs and network processors that provide hardware support for common operations. These solutions have defined the practice of embedded software design and development for the past 25 years or so.

MUCH PROGRESS, LITTLE CHANGE

Given the semiconductor industry's ability to keep pace with Moore's law, the resource limitations of 25 years ago should have almost entirely evaporated by now. Yet embedded software design and development have changed little.

This lack of change may stem from the extreme competitive pressure in products such as consumer electronics, which are based on embedded software and reward only the most efficient solutions. There are, however, many examples where functionality and reliability have proven more important than efficiency, which makes it arguable that factors other than—and possibly even as important as—resource limitations have influenced embedded software's evolution.

Embedded software differs from other software in more fundamental ways. Examining why engineers write embedded software in assembly code or C reveals that efficiency is not their only concern, and may not even be their main one. Reasons for this could include the need to count cycles in a critical inner loop—not to make it fast, but rather to make it predictable.

No widely used programming language integrates a way to specify timing requirements or constraints. Instead, the abstractions they offer focus on scalability—inheritance, dynamic binding, polymorphism, memory management—and, if anything, further obscure timing. Consider, for example, the impact of garbage collection on timing.

Counting cycles becomes extremely difficult on modern processor architectures, where memory hierarchy, dynamic dispatch, and speculative execution make it nearly impossible to tell

how long it will take to execute a particular piece of code. Worse, execution time is context-dependent, which leads to unmanageable variability. Still worse, programming languages usually are Turing complete, which consequently makes execution time undecidable in general.

To get predictable timing, embedded software designers must choose alternative processor architectures such as programmable DSPs, and they must use disciplined programming techniques that, for example, avoid recursion.

Engineers also stick to low-level programming because embedded software typically must interact with hardware specialized to the application. In conventional software, interaction with hardware is the operating system's domain. Typically, application designers do not create device drivers, nor do these drivers form part of an application program. In the embedded software context, however, generic hardware interfaces are rare.

Indeed, higher-level languages do not support creating interfaces to hardware. For example, although concurrency is common in modern programming languages such as Java, which has threads, no widely used programming language includes the notion of interrupts in its semantics. Yet the concept is not difficult and can be built into programming languages. For example, nesC and TinyOS, which are widely used for programming sensor networks, support interrupts at the language level.

Considering these factors, we can see that embedded software engineers do not avoid the many recent improvements in computation out of ignorance. Rather, they seek to avoid a mismatch of the core abstractions and the technologies built upon them.

In embedded software, time matters, yet computing's 20th-century abstractions hold time to be irrelevant. In embedded software, concurrency and interaction with hardware are intrinsic because embedded software engages the physical world in nontrivial ways.

The most influential 20th-century computing abstractions speak only weakly about concurrency, if at all. Even the core 20th-century notion of *computable* is at odds with the requirements of embedded software. In this notion, useful computation terminates, but termination is undecidable. In embedded software, termination is failure—yet to get predictable timing, subcomputations must decidably terminate.

In embedded software, time matters, yet computing's 20th-century abstractions hold time to be irrelevant.

TIMING'S CRUCIAL ROLE

Embedded systems consist of software and hardware integrations in which the software reacts to sensor data and issues commands to hardware actuators.

The physical system forms an integral part of the design, and the software must be conceptualized to operate in concert with it. Physical systems are intrinsically concurrent and temporal. Actions and reactions happen simultaneously and over time, and the metric properties of time play an essential part in the system's behavior.

Prevailing software methods abstract away time, replacing it with ordering. In imperative languages such as C, C++, and Java, the program defines the order of actions, but not their timing.

THE PROBLEM WITH THREADS

Another abstraction, threads or processes, overlays this prevailing imperative abstraction. The operating system typically provides this alternative abstraction, but occasionally the programming language does so.

Threads mainly focus on providing an illusion of parallelism in fundamentally sequential models, and they work well only for modest levels of concurrency or for highly decoupled systems that share resources, where best-effort scheduling policies are sufficient. Indeed, several recent innovative embedded software frameworks, such as The MathWorks' Simulink, UC Berkeley's nesC and TinyOS, and Esterel Technologies' Lustre/SCADE all provide concurrent programming languages with no threads or processes in the programmer's model.

Users generally hold embedded software systems to a much higher reliability standard than general-purpose software. Often, failures in the software can be life threatening. The prevailing concurrency model in general-purpose software does not achieve adequate reliability. This model makes it extremely difficult for humans to understand the interaction between threads. Although we can argue that concurrent computation is inherently complex, threads make it far more so because any part of the system's state can change between any two atomic operations.

The basic techniques for controlling this interaction use semaphores and mutual exclusion locks, methods that date back to the 1960s. Many uses of these techniques lead to deadlock or livelock. In general-purpose computing, this inconvenient event typically forces a program restart or even a reboot.

In embedded software, however, such errors can be far more than inconvenient. Even in general-purpose software systems, interactions with or between device drivers built on these low-level concurrency mechanisms often cause failures. Moreover, developers frequently write software without sufficiently using interlock mechanisms, which results in race conditions that yield nondeterministic program behavior.

In practice, testing cannot easily detect errors from the misuse or nonuse of semaphores and mutual exclusion locks. Code can be exercised for years before a design flaw appears.

Static analysis techniques, such as Sun Microsystems' LockLint, can help, but both conservative approximations and false positives often thwart these methods, thus they are not widely used in practice.

Reliability through clarity

We can argue that multithreaded programs' unreliability stems at least in part from inadequate software engineering processes. For example, better code reviews, specifications, compliance testing, and development process planning can help solve these problems.

Given the difficulty of understanding programs that use threads, however, no amount of process improvement will make such a program reliable if its developers cannot understand it. Formal methods can help detect flaws in threaded programs and, in the process, can improve the designer's understanding of a complex program's behavior. But if the basic mechanisms fundamentally make programs difficult to understand, these improvements will fall short of delivering reliable software.

Prevailing industrial practice in embedded software relies on bench testing for concurrency and timing properties. This has worked reasonably well because programs are small and the software is encased in a box where no outside connectivity can alter its behavior. However, applications today demand that embedded systems be feature-rich and networked, so bench testing and encasing become inadequate.

In a networked environment, it is impossible to test the software under all possible conditions because the environment is unknown. Moreover, general-purpose networking techniques themselves make program behavior much more unpredictable.

REINVENTING COMPUTER SCIENCE

Achieving concurrent and networked embedded software that can be absolutely positively on time—say, to the precision and reliability of digital logic—will, again, require sweeping changes:

- The core abstractions of computing must be modified to embrace time.
- Computer architectures must deliver precisely timed behaviors.
- The hardware–software boundary must be rethought.
- Networking techniques must provide time concurrence.
- Programming languages must embrace time and concurrency in their core semantics.
- Virtual machines must rely less on just-in-time compilation.
- Power management techniques must rely less on voltage and clock speed scaling or must couple these with timing requirements.
- Operating systems must rely less on priorities to indirectly specify timing requirements.
- Memory management techniques must account for timing constraints.
- Complexity theory must morph into schedulability analysis.
- Software engineering methods must change to specify and analyze software's temporal dynamics.
- Developers must rethink the traditional boundary between the operating system and the programming language.

Applications today demand that embedded systems be feature-rich and networked, so bench testing and encasing become inadequate.

In essence, we must reinvent computer science. Fortunately, we have quite a bit of knowledge and experience to draw upon.

Architecture techniques such as software-managed caches promise to deliver much of the benefit of memory hierarchy without the timing unpredictability. Pipeline interleaving and stream-oriented architectures offer deep pipelines with deterministic execution times. FPGAs with processor cores provide alternative hardware and software divisions.

To date, however, all these hardware techniques largely lack programming language and compiler support.

On the software side, operating systems such as TinyOS provide simple ways to create thin wrappers around hardware, and, with nesC, alter the OS/language boundary. Programming languages such as Lustre/SCADE provide understandable and analyzable concurrency. Embedded software languages such as Simulink provide time in their semantics. Bounded pause-time garbage collectors provide memory management with timing determinism.

On the networking side, time-triggered architectures provide deterministic media access and improved fault tolerance. Network time synchronization methods such as IEEE 1588 provide time concurrence at nanosecond resolutions far finer than any processor or software architectures can exploit today.

On the theory side, hybrid systems theory provides a semantics that is both physical and computational.

With so many promising starts, the time is ripe to pull these techniques together and build 21st-century embedded computer science. ■

Edward A. Lee is a professor, chair of the Electrical Engineering Division, and associate chair of Electrical Engineering and Computer Sciences at the University of California, Berkeley. Contact him at eal@eecs.berkeley.edu.

Editor: Wayne Wolf, Dept. of Electrical Engineering, Princeton University, Princeton NJ; wolf@princeton.edu

2005 IEEE Computer Society Professional Membership/Subscription Application

Membership and periodical subscriptions are annualized to and expire on 31 December 2005.
Pay full or half-year rate depending upon the date of receipt by the IEEE Computer Society as indicated below.

Membership Options*

All prices are quoted in U.S. dollars

	FULL YEAR Applications received 16 Aug 04 - 28 Feb 05	HALF YEAR Applications received 1 Mar 05 - 15 Aug 05
1 I do not belong to the IEEE, and I want to join **just** the Computer Society	$ 102 ❑	$51 ❑
2 I want to join **both** the Computer Society and the IEEE:		
I reside in the United States	$195 ❑	$98 ❑
I reside in Canada	$175 ❑	$88 ❑
I reside in Africa/Europe/Middle East	$171 ❑	$86 ❑
I reside in Latin America	$164 ❑	$82 ❑
I reside in Asia/Pacific	$165 ❑	$83 ❑
3 I already belong to the IEEE, and I want to join the Computer Society. (IEEE members need only furnish name, address, and IEEE number with payment.)	$ 44 ❑	$22 ❑

Are you now or were you ever a member of the IEEE?
Yes ❑ No ❑ If yes, provide member number if known: _____

Add Periodicals**

	ISSUES PER YEAR	FULL YEAR Applications received 16 Aug 04 - 28 Feb 05 PRINT	FULL YEAR ELECTRONIC	FULL YEAR COMBO	HALF YEAR Applications received 1 Mar 05 - 15 Aug 05 PRINT	HALF YEAR ELECTRONIC	HALF YEAR COMBO
IEEE Computer Society Digital Library BEST DEAL	n/a	n/a	$118 ❑	n/a	n/a	$59 ❑	n/a
Computing in Science and Engineering	6	$42 ❑	$40 ❑	$55 ❑	$21 ❑	$20 ❑	$28 ❑
IEEE Computer Graphics and Applications	6	$39 ❑	$31 ❑	$51 ❑	$20 ❑	$16 ❑	$26 ❑
IEEE Design & Test of Computers	6	$37 ❑	$30 ❑	$48 ❑	$19 ❑	$15 ❑	$24 ❑
IEEE Intelligent Systems	6	$37 ❑	$30 ❑	$48 ❑	$19 ❑	$15 ❑	$24 ❑
IEEE Internet Computing	6	$39 ❑	$31 ❑	$51 ❑	$20 ❑	$16 ❑	$26 ❑
IT Professional	6	$40 ❑	$32 ❑	$52 ❑	$20 ❑	$16 ❑	$26 ❑
IEEE Micro	6	$37 ❑	$30 ❑	$48 ❑	$19 ❑	$15 ❑	$24 ❑
IEEE MultiMedia	4	$35 ❑	$28 ❑	$46 ❑	$18 ❑	$14 ❑	$23 ❑
IEEE Pervasive Computing	4	$41 ❑	$33 ❑	$53 ❑	$21 ❑	$17 ❑	$27 ❑
IEEE Security & Privacy	6	$41 ❑	$33 ❑	$53 ❑	$21 ❑	$17 ❑	$27 ❑
IEEE Software	6	$44 ❑	$35 ❑	$57 ❑	$22 ❑	$18 ❑	$29 ❑
IEEE/ACM Transactions on Computational Biology and Bioinformatics	4	$35 ❑	$28 ❑	$46 ❑	$18 ❑	$14 ❑	$23 ❑
IEEE/ACM Transactions on Networking†	6	$44 ❑	$33 ❑	$55 ❑	$22 ❑	$17 ❑	$28 ❑
IEEE Transactions on:							
Computers	12	$41 ❑	$33 ❑	$53 ❑	$21 ❑	$17 ❑	$27 ❑
Dependable and Secure Computing	4	$31 ❑	$25 ❑	$40 ❑	$16 ❑	$13 ❑	$20 ❑
Information Technology in Biomedicine†	4	$45 ❑	$35 ❑	$54 ❑	$23 ❑	n/a	$27 ❑
Knowledge and Data Engineering	12	$43 ❑	$34 ❑	$56 ❑	$22 ❑	$17 ❑	$28 ❑
Mobile Computing	6	$32 ❑	$26 ❑	$42 ❑	$16 ❑	$13 ❑	$21 ❑
Multimedia†	6	n/a	n/a	$40 ❑	n/a	n/a	n/a
NanoBioscience†	4	$40 ❑	$30 ❑	$48 ❑	$20 ❑	n/a	$24 ❑
Parallel and Distributed Systems	12	$40 ❑	$32 ❑	$52 ❑	$20 ❑	$16 ❑	$26 ❑
Pattern Analysis and Machine Intelligence	12	$44 ❑	$35 ❑	$57 ❑	$22 ❑	$18 ❑	$29 ❑
Software Engineering	12	$38 ❑	$30 ❑	$49 ❑	$19 ❑	$15 ❑	$25 ❑
Visualization and Computer Graphics	6	$34 ❑	$27 ❑	$44 ❑	$17 ❑	$14 ❑	$22 ❑
VLSI Systems†	12	n/a	n/a	$28 ❑	n/a	n/a	$14 ❑
IEEE Annals of the History of Computing	4	$31 ❑	$25 ❑	$40 ❑	$16 ❑	$13 ❑	$20 ❑

Choose PRINT for paper issues delivered via normal postal channels.
Choose ELECTRONIC for 2005 online access to all issues published from 1988 forward.
Choose COMBO for both print and electronic.

Payment Information

Payment required with application

Membership fee $ _____
Periodicals total $ _____
Applicable sales tax*** $ _____
Total $ _____

Enclosed:
❑ Check/Money Order****

Charge my:
❑ MasterCard ❑ Visa
❑ American Express ❑ Diner's Club

_____
Card number

_____
Expiration date (month/year)

_____
Signature

USA-only include 5-digit billing zip code
☐☐☐☐☐

* Member dues include $19 for a 12-month subscription to *Computer*.

** Periodicals purchased at member prices are for the member's personal use only.

*** Canadian residents add 15% HST or 7% GST to total. AL, AZ, CO, DC, NM, and WV add sales tax to all periodicals. GA, IN, KY, MD, and MO add sales tax to print and combo periodicals. NY add sales tax to electronic and combo periodicals. European Union residents add VAT tax to electronic periodicals.

**** Payable to the IEEE in U.S. dollars drawn on a U.S. bank account. Please include member name and number (if known) on your check.

† Not part of the IEEE Computer Society Digital Library. Electronic access is through www.ieee.org/ieeexplore.

For fastest service, apply online at www.computer.org/join

NOTE: In order for us to process your application, you must complete and return BOTH sides of this form to the office nearest you:

Asia/Pacific Office
IEEE Computer Society
Watanabe Bldg.
1-4-2 Minami-Aoyama
Minato-ku, Tokyo 107-0062 Japan
Phone: +81 3 3408 3118
Fax: +81 3 3408 3553
E-mail: tokyo.ofc@computer.org

Publications Office
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1314 USA
Phone: +1 800 272 6657 (USA and Canada)
Phone: +1 714 821 8380 (worldwide)
Fax: +1 714 821 4641
E-mail: help@computer.org

Allow up to 8 weeks to complete application processing. Allow a minimum of 6 to 10 weeks for delivery of print periodicals.

Personal Information

Enter your name as you want it to appear on correspondence. As a key identifier in our database, circle your last/surname.

Male ❑ Female ❑ Date of birth (Day/Month/Year) _____

Title | First name | Middle | Last/Surname

Home address

City | State/Province

Postal code | Country

Home telephone | Home facsimile

Preferred e-mail

Send mail to: ❑ Home address ❑ Business address

Educational Information

First professional degree completed | Month/Year degree received

Program major/course of study

College/University | State/Province | Country

Highest technical degree received | Program/Course of study

Month/Year received

College/University | State/Province | Country

Business/Professional Information

Title/Position

Years in current position | Years of practice since graduation

Employer name | Department/Division

Street address | City | State/Province

Postal code | Country

Office phone | Office facsimile

I hereby make application for Computer Society and/or IEEE membership and agree to be governed by IEEE's Constitution, Bylaws, Statements of Policies and Procedures, and Code of Ethics. I authorize release of information related to this application to determine my qualifications for membership.

Signature | Date

APPLICATION MUST BE SIGNED

NOTE: In order for us to process your application, you must complete and return both sides of this form.

IEEE

IEEE COMPUTER SOCIETY

BPA Information

This information is used by society magazines to verify their annual circulation. Please refer to the audit codes and indicate your selections in the box provided.

A. Primary line of business

1. Computers
2. Computer peripheral equipment
3. Software
4. Office and business machines
5. Test, measurement and instrumentation equipment
6. Communications systems and equipment
7. Navigation and guidance systems and equipment
8. Consumer electronics/appliances
9. Industrial equipment, controls and systems
10. ICs and microprocessors
11. Semiconductors, components, sub-assemblies, materials and supplies
12. Aircraft, missiles, space and ground support equipment
13. Oceanography and support equipment
14. Medical electronic equipment
15. OEM incorporating electronics in their end product (not elsewhere classified)
16. Independent and university research, test and design laboratories and consultants (not connected with a manufacturing company)
17. Government agencies and armed forces
18. Companies using and/or incorporating any electronic products in their manufacturing, processing, research, or development activities
19. Telecommunications services, telephone (including cellular)
20. Broadcast services (TV, cable, radio)
21. Transportation services (airlines, railroads, etc.)
22. Computer and communications and data processing services
23. Power production, generation, transmission, and distribution
24. Other commercial users of electrical, electronic equipment and services (not elsewhere classified)
25. Distributor (reseller, wholesaler, retailer)
26. University, college/other education institutions, libraries
27. Retired
28. Others (allied to this field) _____

B. Principal job function

1. General and corporate management
2. Engineering management
3. Project engineering management
4. Research and development management
5. Design engineering management - analog
6. Design engineering management - digital
7. Research and development engineering
8. Design/development engineering - analog
9. Design/development engineering - digital
10. Hardware engineering
11. Software design/development
12. Computer science
13. Science/physics/mathematics
14. Engineering (not elsewhere classified)
15. Marketing/sales/purchasing
16. Consulting
17. Education/teaching
18. Retired
19. Other _____

C. Principal responsibility

1. Engineering or scientific management
2. Management other than engineering
3. Engineering design
4. Engineering
5. Software: science/management/engineering
6. Education/teaching
7. Consulting
8. Retired
9. Other _____

D. Title

1. Chairman of the Board/President/CEO
2. Owner/Partner
3. General Manager
4. V.P. Operations
5. V.P. Engineering/Director Engineering
6. Chief Engineer/Chief Scientist
7. Engineering Manager
8. Scientific Manager
9. Member of Technical Staff
10. Design Engineering Manager
11. Design Engineer
12. Hardware Engineer
13. Software Engineer
14. Computer Scientist
15. Dean/Professor/Instructor
16. Consultant
17. Retired
18. Other Professional/Technical _____

Developer-Focused Assurance Requirements

Gary Stoneburner, Johns Hopkins University/Applied Physics Laboratory

The ISO/IEC *Common Criteria for Information Technology Security Evaluation* could benefit from focusing on development, rather than evaluation, to provide assurance.

In 1999, the International Organization for Standardization and the International Electrotechnical Commission jointly published the *Common Criteria for Information Technology Security Evaluation* (ISO/IEC 15408, Oct. 1999; www.commoncriteriaportal.org) to provide IT security evaluation guidelines that extend to an international community.

The assurance requirements, including prepackaged sets of Evaluation Assurance Levels (EALs) in the Common Criteria (CC), represent the paradigm that assurance equals evaluation, and more evaluation leads to more assurance.

This paradigm is at odds with the commercial off-the-shelf (COTS) marketplace, neither reflecting how confidence is typically achieved nor providing a cost-effective means for supplying grounds for confidence in the security capabilities of the information technology being evaluated.

ASSURANCE FUNDAMENTALS

The Common Criteria document defines assurance as "grounds for confidence that an entity meets its security objectives." Despite this internationally recognized definition, in practice the term has multiple, conflicting meanings.

In IT security contexts, "assurance" describes the degree to which confidence is held, the amount of information available upon which to base confidence, and the characteristics of the technology that exist, whether or not anyone exhibits confidence in them. The IT security community uses "assurance" as

- a measure of a subjective human attribution: confidence,
- an objective measurement of or fact about the IT system, or
- an IT characteristic that exists independent of confidence in the system or any measurement of or fact about the system.

Here I will use the CC definition, paraphrased as, "the grounds for confidence that the IT meets explicitly identified security expectations."

Assurance as an objective measurement of or fact about your IT system, upon which you can base your confidence, primarily concerns what is known about the system. Another trait frequently associated with assurance, security quality, describes the inherent system characteristic that is the object of assurance and in which users place their confidence.

Although confidence is subjective, the system characteristic of security quality exists independent of user confidence. Users can have high confidence despite low quality or low confidence even though the quality is high.

MEETING SECURITY OBJECTIVES

Effective security requirement sets will contain or point to the following:

- a nontechnical, clear, and concise description of the nature of the operational environment in business or mission terms and the degree of protection this requirement set addresses;
- a set of security requirements to meet this description; and
- a compelling rationale for the claim that the given requirements will meet the needs.

When those who make purchase decisions embrace a set of effective criteria as part of their needs statement, the criteria can be considered useful. Criteria are not useful if they lack effectiveness, do not enjoy user support, or enjoy user support but not support from those who make purchasing decisions.

EFFECTIVE ASSURANCES

Effective assurance provides evidence that

- *Inspires confidence.* If the assurance does not inspire confidence in the IT system, the evidence is not useful except perhaps to highlight that confidence is not warranted.

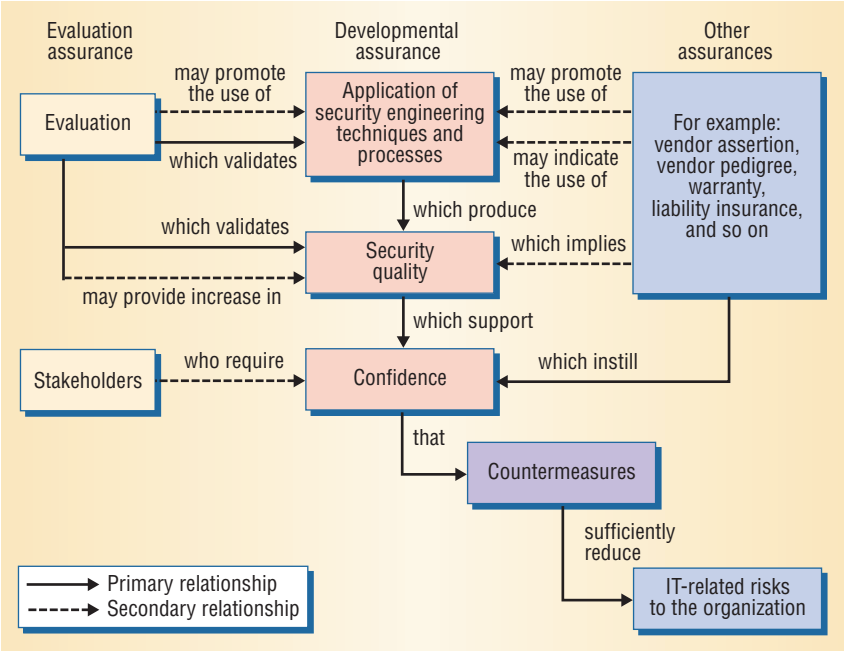


Figure 1. Assurance model, displaying the relationship between various assurance means, security quality, and confidence.

- *Fosters objectivity*. Scientifically based evidence provides a foundation for the inherently subjective decision of whether or not to bestow confidence.
- *Is cost-effective*. The relative usefulness and the degree of objective, new information obtained must be commensurate with the cost to obtain it.

Figure 1 captures these assurance concepts.

DEVELOPER-FOCUSED ASSURANCE

Changing current CC components in the following ways would encourage a focus on development instead of evaluation:

- *Overall focus*. The CC should move from a mindset of "documentation for evaluation" to an explicit statement describing developer actions that enhance the security capability.
- *Developer actions*. Rather than leaving many developer actions to be defined by implication, all should be explicitly stated in the developer action elements.
- *Evaluator actions*. Instead of describing evaluator actions in terms of the evidence used, the document should state what the evaluator is to determine.
- *Differentiation between assurance levels*. Rather than defining assurance levels by the amount of evaluation performed, the definition should describe the developer actions and the resulting potential for increased security quality.

This shift places more responsibility for building confidence in the hands of IT developers.

PROPOSED ENHANCEMENTS

This example shows proposed enhancements to the current CC document, using the configuration management component ACM_CAP.1, "Version numbers."

Current CC: Evaluator focus

The CC assurance elements for ACM_CAP.1 are:

Developer action

- ACM_CAP.1.1D: "The developer shall provide a reference for the TOE [Target of Evaluation]."

Content and presentation of evidence

- ACM_CAP.1.1C: "The reference for the TOE shall be unique to each version of the TOE."
- ACM_CAP.1.2C: "The TOE shall be labeled with its reference."

Evaluator action

- ACM_CAP.1.1E: "The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence."

Proposed: Developer focus

Under the proposed schema, these elements would change to highlight the developer's role:

Developer action

- ACM_CAP-L.1.1D: "The developer shall label the TOE with a unique reference for each version."

Content and presentation of evidence

- None.

Evaluator action

- ACM_CAP-L.1.1E: "The evaluator shall check that the TOE is labeled with a reference that can reasonably be expected to be unique to each version."

ASSURANCE PACKAGES—A SUGGESTION

As an extension of this developer-focused approach, an alternative set of assurance levels could potentially replace the EALs of the CC. These nine assurance levels—AL1 through AL9—build on concepts already present in the current CC document.

AL4, AL6, and AL8 call for less third-party assessment and would apply when

- a vendor has proven trustworthy in being able and likely to meet

requirements at this assurance level, or
- the marketplace will not support the cost of an extensive independent evaluation—and therefore will accept the resulting added risks.

AL5, AL7, and AL9 call for extensive third-party assessment and would apply when

- a vendor has not proven trustworthy in being able and likely to meet requirements at this assurance level, and
- the marketplace will support the costs associated with an extensive independent evaluation, not accepting the risk associated with greater reliance on vendor assertion.

AL1—Vulnerability testing. This basic level of assurance would rely on third-party testing to look for obvious flaws and common vulnerabilities. Since this level doesn't require security quality, cost-effectiveness dictates minimal added cost from evaluator actions.

AL2—Functional testing. An increase in assurance over AL1, AL2 adds evaluator verification of functional compliance with the intention that if developer testing is adequate, review of that testing can be adequate for evaluator verification. Since AL2 still would not require security quality, cost-effectiveness would continue to dictate minimal added cost from evaluator actions.

AL3—Rigorous COTS development. This level would provide greater assurance primarily through developer actions such as positive security engineering and sound development practices. Achieving the required depth and rigor of these actions should not require changes to existing good commercial practices. Because significant security engineering—and hence higher security quality—would not yet be required, the evaluation activities remain minimal. This level resembles CC EAL3, but with less evaluation.

AL4—Significant security engineering. This level would aim to produce significantly higher security quality by requiring rigorous commercial development practices, coupled with significant application of specialist security engineering knowledge and skills. For the most part, developer documentation would be taken at face value. This is similar to CC EAL5, but with less evaluation.

AL5—Verified significant security engineering. Adding extensive third-party evaluation to AL4 would produce verified, moderate-quality components. The resulting set of assurances would be similar to CC EAL5.

AL6—Rigorous security engineering. By adding rigorous development processes and security engineering to AL4, this level would produce high-assurance components. This has similarities to CC EAL6, but with less evaluation.

AL7—Verified, rigorous security engineering. Adding extensive third-party validation to AL6 would produce verified, high-assurance components. AL7 is similar to CC EAL6.

AL8—Formal methods. This level adds formal methods to AL6, producing very-high-assurance components. It resembles CC EAL7, but with less evaluation.

AL9—Verified formal methods. By adding extensive third-party validation to AL8, this level would produce verified, very-high-assurance components, similar to CC EAL7.

An information technology system can withstand attack not on the basis of how much an evaluator measures, but only if what the developer builds is robust. Trying to insert quality through inspection is now known to be a flawed approach. As management guru W. Edwards Deming stated, "You cannot inspect quality into the product; it is already there."

Redirecting assurance activities toward the development of more trustworthy information technology requires a paradigm shift. Creating a new set of assurances such as those proposed here would redirect the focus for producing security quality back to what developers should do. It would also emphasize measuring the results only as a means of verifying the quality.

The Software Engineering Proverbs Web site (www.multicians.org/thvv/proverbs.html) quotes Albert Einstein as saying, "The significant problems we face cannot be solved by the same level of thinking that created them." With that in mind, I invite you to enter the dialogue on developer-focused assurances taking place through the listserv e-mail list at Developer-Actions-L@listserv.jhuapl.edu. ■

***Gary Stoneburner** is a member of the senior professional staff at the Johns Hopkins University's Applied Physics Laboratory. Contact him at Gary.Stoneburner@jhuapl.edu.*

Editor: Jack Cole, US Army Research Laboratory's Information Assurance Center, jack.cole@ieee.org; http://msstc.org/cole.

The Tech Buzz Game

Bernard Mangold, Mike Dooley, Gary W. Flake, Havi Hoffman, Tejaswi Kasturi, and David M. Pennock, Yahoo! Research Labs
Rael Dornfest, O'Reilly Media

A Yahoo!/O'Reilly fantasy prediction market forecasts high-tech events and trends.

Prediction markets, also known as information or decision markets, are designed to forecast future events or trends. In such markets the payoff is tied to some outcome, such as an election result, and at any given time the trading price reflects traders' consensus on the outcome's likelihood.

Numerous prediction markets operate online. For example, Iowa Electronic Markets (www.biz.uiowa.edu/iem) are real-money political and economic futures markets, while TradeSports (www.TradeSports.com) is a sports-related real-money exchange. Popular play-money or fantasy markets include the Hollywood Stock Exchange (www.hsx.com) and the Foresight Exchange (www.ideosphere.com).

Internet-based prediction markets can easily aggregate the insights of an unlimited number of potentially knowledgeable people asynchronously. Researchers studying these markets in recent years have found them to be remarkably accurate.

THE TECH BUZZ GAME

The Tech Buzz Game (http://buzz.research.yahoo.com)—a joint venture between Yahoo! Research Labs and O'Reilly Media—is a fantasy prediction market launched in March 2005 at the O'Reilly Emerging Technology (ETech) Conference in San Diego, California.

The game consists of multiple submarkets that pit a handful of rival technologies, each represented by a stock, against one another. For example, the Browser Wars market contains seven stocks: Internet Explorer, Firefox, Opera, Mozilla, Camino, Konqueror, and Safari. Players have access to the current "buzz" around each technology, as measured by the number of Yahoo! Search users seeking information on it.

The game's object is to anticipate future search buzz and buy and sell stocks accordingly. Thus, a player who believes BitTorrent stock is undervalued might buy shares, while a player who thinks BitTorrent is overpriced might sell the stock or instead purchase shares in a competing peer-to-peer technology.

Research goals

The Tech Buzz Game serves two key research-oriented goals. One is to evaluate the power of prediction markets to forecast high-tech trends. O'Reilly Media (http://radar.oreilly.com) designed the game's ontology based on the landscape of technologies, products, and companies on its radar.

The other goal of the Tech Buzz Game is to field test the *dynamic pari-mutuel market*, a Yahoo! Research Labs trading mechanism designed to price and allocate shares. The "Dynamic Pari-Mutuel" sidebar describes this mechanism in more detail.

Buzz scores

Each stock in Tech Buzz Game is associated with a number of buzz words, or search phrases. For example, buzz words for Internet Explorer include "ie," "internet explorer," and "ie6 download." The game uses Yahoo! Search (http://search.yahoo.com) to generate a seed set of buzz words and then uses Yahoo! Search Web Services (http://developer.yahoo.net) to expand the set.

A stock's *buzz score* is the number of searches of buzz words over the past seven days, as a percentage of all searches in the same market. Thus, if searches for Internet Explorer make up 60 percent of all Yahoo! searches in the Browser Wars market, then IE's buzz score is 60. The buzz scores of all technologies within a market always add up to 100.

The buzz-scoring methodology was originally developed for the Yahoo! Buzz Index (http://buzz.yahoo.com), which tracks Web search spikes and trends.

Trading interface

Software developed by NewsFutures (http://NewsFutures.com) powers the game. Players enter the amount of

Published by the IEEE Computer Society

money they want to invest in a specific stock, and the system computes how many shares they're entitled to in return for their investment. Players don't need to deal with separate bid-and-ask queues or wait for a counterparty to execute a trade.

However, the total cost for all shares isn't equal to the current price multiplied by the number of shares because not all shares are purchased at the current price. Instead, as players purchase more shares, the price increases continuously. Each share purchased thus costs a little more than the previous one.

Selling is likewise mechanically simple: Players enter the number of shares they wish to sell, and the system computes their proceeds. Again, as each additional share is sold, the price decreases continuously. Thus, each incremental share sold returns a little less money than the previous share, and the total proceeds are less than the current price multiplied by the number of shares.

Interest in a technology ultimately determines its stock value. For example, Figure 1 graphs the prices, trading volume, and buzz scores for Wi-Fi and WiMax, the two competing stocks in the Wireless Internet market. In mid-April 2005, WiMax's buzz began to grow relative to Wi-Fi, following announcements of new WiMax chips from Intel and Fujitsu as well as a WiMax deployment partnership between Intel and Sprint.

Price changes appear to parallel changes in buzz scores, but graphical analysis alone is insufficient to judge which is leading the other. We plan to conduct statistical analyses across all Tech Buzz Game markets to examine the hypothesis that prices anticipate buzz trends.

Dividends and cash-out events

Paid dividends and the final cash settlement are in proportion to actual search buzz. Thus, savvy traders don't engage in a "beauty contest" of picking their favorite technologies but consider both prices and buzz scores, buying stocks where prices appear low compared to expected future buzz scores.

On every Friday at 6:00 pm Eastern time, each stock receives a total dividend equal to 100 times the stock's buzz score. For example, if IE's buzz score is 60, then the total dividend for IE is $6,000. The total dividend is then

Dynamic Pari-Mutuel

The dynamic pari-mutuel determines the way share prices in the Tech Buzz Game change by using a continuous function solution to a set of differential equations. The game attempts to combine the advantages of two common market mechanisms: the continuous double auction (CDA), which is used in stock exchanges such as Wall Street, and the pari-mutuel, which is used for betting on horse races and other sporting events.

Because a CDA lets you buy shares in a stock only if someone else is willing to sell them at a price less than or equal to what you're willing to pay, the stock's market price reflects current demand. Thus, if you correctly anticipate that demand will increase, you can realize a profit by buying the stock and then selling it after increased demand has raised its price. When there are few traders, however, the best buy and sell prices can be far apart—a situation known as the *thin market problem*.

The pari-mutuel avoids this problem but doesn't allow you to profit by predicting future demand. For example, after a horse race, the money the ticket holders spent on all the tickets they bought is divided in proportion to the number of winning tickets they own. The problem with this mechanism is that there's no incentive to buy early; in fact, the best strategy is to wait until the last possible moment to buy.

The dynamic pari-mutuel avoids the thin market problem while ensuring that stock prices reflect demand. It functions like a classic pari-mutuel in that you can always make purchases for each outcome, but it resembles a CDA in that prices increase with demand.

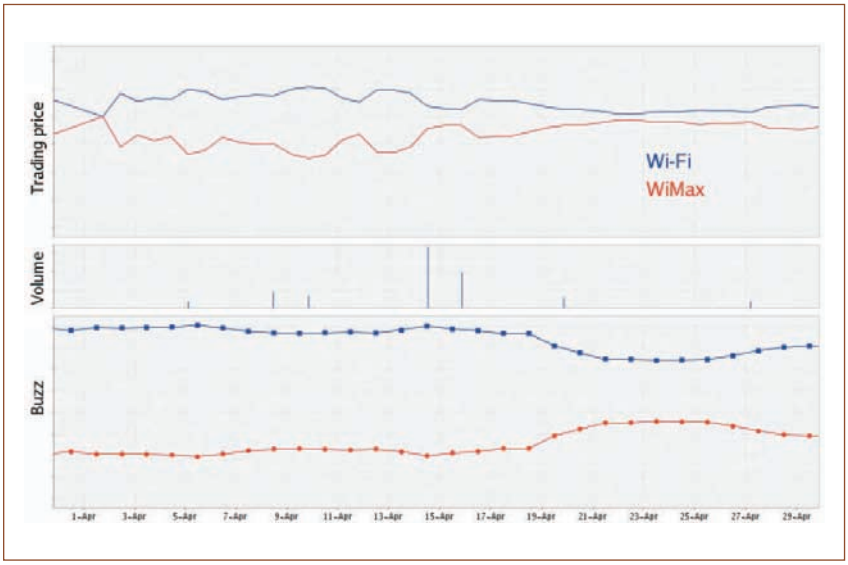


Figure 1. Stock value. In the Wireless Internet market, Wi-Fi and WiMax price changes appear to parallel changes in buzz scores.

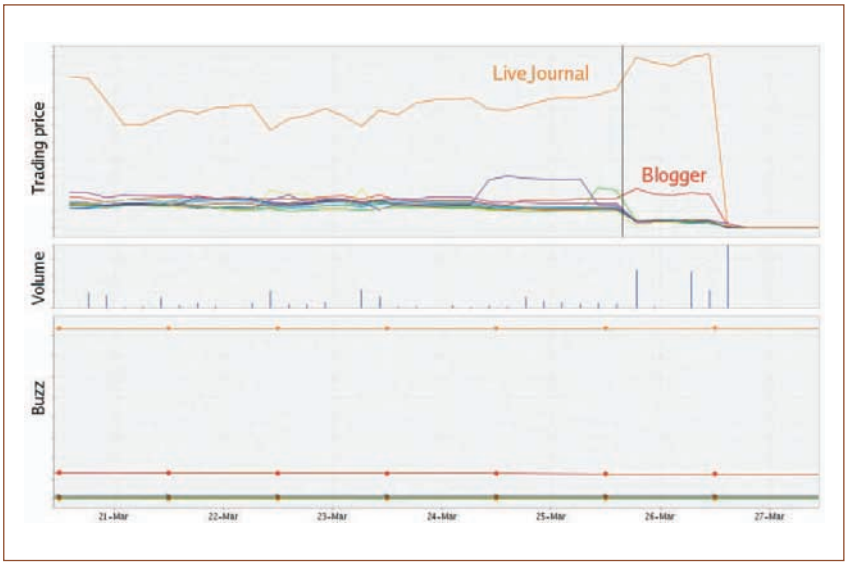


Figure 2. Example of market collapse. A flaw in the dynamic pari-mutuel's money-ratio price function encouraged arbitrage, causing stock prices to fall precipitously.

distributed to shareholders, with each share receiving an equal portion.

Shares are liquidated for cash at long-term intervals. During a cash-out event, all money in every market is distributed to shareholders. Within each market, all money is first allocated among stocks according to buzz scores. For example, if the buzz scores for IE and Firefox are 60 and 20, respectively, and the total Browser Wars market capitalization is $100,000, then $60,000 will be allocated to IE and $20,000 to Firefox. The money allocated to each stock is then distributed to shareholders, with each share receiving an equal portion.

OUT OF THE GATES

The Tech Buzz Game got off to a fast and furious start. The ETech crowd is decidedly alpha geek, with a huge density of wirelessly connected gadget-wielding bloggers. Many conference attendees signed on to play the game as soon as it was announced and immediately began plotting strategies to help them win.

In the first week, the game site received 2.7 million hits. Participants activated 13,310 accounts, placed 117,530 orders, invested $88,480,577 fantasy dollars, and purchased 46,379,241 shares.

A core group of dedicated players demonstrated impressive creativity in building supporting tools and bots, deriving equations, and debating strategies. One player started an independently moderated Yahoo! group for players to discuss the game outside the official message boards.

These forums often resemble typical stock market message boards—full of "pump and dump" exhortations, junk advice, flame wars, and rants. Those who read the message boards with care, separating rare good advice from the noise, fare well. Many players, especially novices, simply buy the technologies they like regardless of price. These players tend to lose money, then either learn to invest more wisely or give up.

Many players expend equal energy subverting the rules. From the beginning, the game was inundated with cheaters who opened up hundreds of accounts, orchestrating them to artificially inflate the stocks in their main portfolio. Consequently, we now employ e-mail verification and CAPTCHA (www.captcha.net) controls, log IP addresses, and search the database for signs of suspect coordination of transactions.

In addition to being a useful research tool, the game is a fascinating social experiment, complete with archetypes like the Leader, the Lurker, the Cheater, the Braggart, and the Novice. Some successful players openly share their strategies, data, tools, and analyses, while others try to cheat their way to the top. One player openly boasted of illicitly amassing a fortune and announced when and where he was going to invest his ill-gotten gains.

DOWN THE STRETCH

During the Tech Buzz Game's second week, stock prices in many markets began falling precipitously, often below their initial starting value, with no signs of stopping. This initially baffled us given that each market was designed to be a zero-sum game in which one stock's price fall would cause other stocks' prices to rise.

The behavior resulted from a flaw in the dynamic pari-mutuel's *money-ratio* price function, which defines the ratio of any two stock prices in the same market as always equal to the ratio of money invested in the stocks. This function enabled traders to perform arbitrage via a four-step process: buy a cheap stock, buy an expensive stock, sell the cheap stock, and sell the expensive stock. If conditions are right, the sequence produces a net positive gain.

Through a combination of mathematical study and trial and error, two 17-year-old students uncovered the flaw. Other traders caught on and exploited it rapidly, causing all stocks in some markets to drop toward zero. Figure 2 shows an example of one such collapse in the Weblog Applications and Services market.

Our initial response was to disallow the purchase of multiple stocks in the same market, which makes arbitrage by any single player impossible. However, this Band-Aid fix stemmed the tide for less than a day. Soon pairs of players colluded to carry out arbitrage in tandem, sharing the spoils. Several even

actively sought out partners in crime in the Yahoo! group chat room.

We employed a more permanent fix by replacing the money-ratio price function with a *share-ratio* price function that defines the ratio of any two prices in a market as always equal to the ratio of outstanding shares for those two stocks. For example, if the number of outstanding shares of IE is twice that of Firefox, then IE's price is twice that of Firefox; if the number of shares is equal, the prices are equal.

The share-ratio price function does not admit arbitrage. The dynamic pari-mutuel mechanism has been running fairly smoothly since the change.

Given the interest among programmers in the Tech Buzz Game, we recently implemented a Representational State Transfer application programming interface to open up access to third-party programs. REST accepts queries as URLs and returns results in easily parseable XML.

Currently, the service is read-only: Users can retrieve stock prices, buzz scores, number of shares outstanding, and market cap information. This makes it possible to create simple support applications such as stock tickers, RSS feeds, or "triggers" that inform users about events such as a major price or buzz score change.

In the future, players will be able to use REST to access their account information and make trades. They will be able to write an application that fully replaces the existing Web-based interface, so that they won't even need to visit the site to play the game. We also plan to expand the Tech Buzz Game to include more markets and areas as well as support tools for use in defining markets and stocks. ■

Bernard Mangold *is the senior director of Yahoo! Research Labs. Contact him at mangoldb@yahoo-inc.com.*

Mike Dooley *is a senior developer at Yahoo! Research Labs. Contact him at dooleym@yahoo-inc.com.*

Gary W. Flake*, a distinguished engineer at Microsoft Corp., contributed to this work as head of Yahoo! Research Labs. Contact him at gary.flake@usa.net.*

Havi Hoffman *is a writer and editor at Yahoo!. Contact her at havi@yahoo-inc.com.*

Teijaswi Kasturi *is a developer at Yahoo! Research Labs. Contact him at kasturit@yahoo-inc.com.*

David M. Pennock *is a senior research scientist at Yahoo! Research Labs. Contact him at pennockd@yahoo-inc.com.*

Rael Dornfest *is chief technology officer at O'Reilly Media. Contact him at rael@oreilly.com.*

Editor: Richard G. Mathieu, Dept. of Decision Sciences and MIS, St. Louis Univ., St. Louis, MO; mathieur@slu.edu

Figure courtesy of Matthias Höchsmann, Björn Voss, and Robert Giegerich.

The Turning of the Wheel

Neville Holmes, University of Tasmania

Computing has, according to some recent popular articles, found something wonderfully new—*virtualization*.

Reading some of these articles, two thoughts struck me. First, that the word must be one of the ugliest and most awkward to be introduced recently and, second, that virtuality in digital technology is far from new.

Indeed, if we take written language as the second great digital technology, as we should, the scribes in the scriptoria of old Europe were virtual authors. More recently, in the early 1970s I worked interactively through the Cambridge Monitor System interface on a virtual System/360 computer provided by a hypervisor called CP67.

Three sources spurred me to draft this column: an April *Computer* article titled "Overcoming the Internet Impasse through Virtualization" (T. Anderson et al., pp. 34-41), a column in the same issue on the oxymoronic topic *virtual reality*, and *Computer*'s entire May issue dedicated to virtuality but with only high-level consideration of its history.

TIME SHARING

In the 1960s, computing manufacturers offered two kinds of machines:

- commercial, usually with decimal arithmetic; and
- scientific, usually with binary arithmetic.

Early digital computers had operators who ran individual programs when the programmers didn't. Given the machines' great expense, developers sought ways to automate their operation and increase their throughput.

These early methods combined job stacking—the automatic transfer of control from one program to the next with peripheral transcription—and the use of a cheap machine to transfer data between the fast magnetic tape used by the mainframe and the slow punched card and paper-tape machines and printers.

These batching methods reached their culmination in the mid-1960s with the somewhat chaotic but eventually successful introduction of IBM's System/360 operating systems, which used multiprogramming and SPOOLing (Simultaneous Peripheral Operations OnLine).

Because the 360 architecture combined binary and decimal arithmetic, IBM planners had imagined that their product would be as suitable for the scientific world as for the commercial. This prediction proved wildly inaccurate.

In the business world, management could dictate that users be kept at arm's length and programmers be banned from machine rooms, but in the scientific world, users ruled the roost. Users are more interested in getting good results than in keeping costs down. Scientific users, who often programmed for themselves, found the extra control in using the machines attractive. This led to the idea of time-sharing.

In its basic form, time-sharing relied on most users being at their Teletype terminals mulling over what happened last and what to do next, so that a ready-to-run user's program could take over the machine temporarily. In practice, time-sharing required virtual memory to be successful.

The new in digital technology is not what we do, only what we do it with.

By the time IBM finally got its 360 batch operating systems up and running, time-sharing had established itself, particularly in universities, and it looked as if the company would lose a huge market. In response, IBM mounted two massive projects, one in Poughkeepsie and another in Mohansic.

The Poughkeepsie project, Time-Sharing Option, aimed to provide time-sharing as a subsystem of the top-of-the-line MVT operating system. TSO distracted developers greatly from the basic improvements that system needed, got off the ground slowly, and was not very successful.

The other project, a time-sharing system called TSS/360, built on IBM's experience in collaborating with universities on their time-sharing projects. IBM intended this system to run on a special 360, the Model 67.

TSS proved a complete flop. Many 360/67 users outside IBM switched to the Michigan Terminal System. When I joined IBM Australia's Systems Development Institute in late 1970, the

Continued on page 98

Continued from page 100

Institute was abandoning TSS for CP67/CMS. This is where the virtual machines come in.

VIRTUAL MACHINES

Tom Van Vleck briefly describes the CP67/CMS's development (www.multicians.org/thvv/360-67.html), while Melinda Varian does so in delightful detail (pucc.princeton.edu/~melinda/25paper.pdf).

Early time-sharing adopted the idea of providing concurrent constrained use of the computer by users' programs.

A relatively small team at IBM's Cambridge Scientific Center in Massachusetts covertly implemented the forerunner of CP67/CMS, CP-40/CMS on a modified 360/40. They used the term *virtual machine*, having heard of it being used for an earlier, more conventional, IBM time-sharing system.

Their elegant work, starting in the last week of 1964, was inspired by the idea that they would provide for each user a strictly virtual machine indistinguishable from a real one by a user program. After the 360/67 intended for TSS/360 was announced, they converted their work to run on that machine.

The strict virtual machine had many advantages. The hypervisor or CP turned out to be relatively simple to do. Because all time-sharing users had their own "machines," with their own disk partitions, the CMS only had to support a single user and had simple support requirements compared to the requirements for conventional time-sharing.

Because the virtual machine was strict, we ran the ordinary OS/360 in a virtual machine for our batch work in Canberra. We even tested new versions of CP in a virtual machine.

Several observations spring from this earlier development. First, the term virtual machine had a specific meaning, and its use today is degenerate. Of course, all computing is virtual, but we should use technological terminology to enhance meaning, not remove it. For example, the Java virtual machine is properly a simulator, or, if done in hardware, an emulator. The use of emulation ensured the success of the System/360 machines.

More generally, the epidemic use of the adjective *virtual* is akin to the epidemic use of *user friendly* in the 1970s. Further, it's a pity that the pathetic initialism VMM has been adopted for what had once been more expressively and comfortably called a *hypervisor*.

The epidemic use of the adjective *virtual* is akin to the epidemic use of *user friendly* in the 1970s.

Second, in a large company, people make technical decisions for political reasons. Cloistered development managers in IBM rejected the virtual machine and sought to get rid of it long after it had saved the day for them. The computer industry today would have been quite different if IBM managers had enthusiastically adopted the principle once circumstance forced it on them.

MULTICORE CHIPS

It's probably simple fashion that couples multicore chips with support for virtual machines ("Chip Makers Turn to Multicore Processors," D. Geer, *Computer*, May 2005, pp. 11-13). Certainly, strict virtual machines need hardware support, but multiprocessing has nothing directly to do with that.

Multicore chip development seems to have happened because the manufacturers of these chips have run out of ideas for using all the circuitry their improved manufacturing methods have made available. I would rather they had provided improved interval and complex arithmetic, or support for console windowing independently of the operating system. This would help foster the reportedly growing movement back to thin clients, known as dumb terminals in the 1980s.

Multicore chip development has two interesting aspects. First, consider the licensing issue mentioned in the *Computer* news item. If a chip has two processors using proprietary software, how many license fees must be paid? Second, continuing development raises a question: If multiprocessor chips succeed in the market, what then? More cores per chip seem likely.

How then will software adapt to these changing architectures? Another wheel could come full circle, one called *strict virtual architecture*.

VIRTUAL ARCHITECTURE

A little-known wild-duck IBM project illustrates what I mean by strict virtual architecture: the System/38.

Universality was the official marketing story for the IBM/360: one architecture to rule the world, with the 360 being the number of degrees to a full circle. But IBM soon branched out into a spectrum of incompatible architectures—Series/1, System/3, System/7, System/32, and System/34, for example. Developers typically used these architectures for problems and customers too small to warrant a System/360 with its accompanying data processing department.

Eventually, IBM started a project in Rochester, Minnesota, to bring the small commercial machine architectures together. Their System/38 virtual architecture can best be described as glorious. Instruction addresses referred to objects so that, for example, the program had only two add instructions, one with two addresses and one with three. The three-address add could, for example, add a packed decimal value to a binary value and produce a character result. All objects were stored in a 64-bit address space, even though early machines used only 48 bits.

Disk storage supported the address space but could not be directly used from programs. For a long time, IBM supported only Cobol and RPG, but no assembler. Their code was compiled to instructions in the virtual architecture. The objects the compilers pro-

duced could not be run directly, however. The operating system used a machine instruction to convert a compiled object to a runnable object, at which point the virtuality appeared.

Creating the runnable program involved a translation from the virtual instruction set into the actual instruction set. This translation made the highly sophisticated object-oriented virtual instruction set possible, and this set hid the complexities of the actual machine from the programmer.

If an improved actual machine was required, a new *create program* instruction could be written. Making old programs run on the new machine would require only the creation of a new runnable program from the old compiled program objects.

All of which makes it seem that the best way to cope with the developing multicore chips would be to have a strict virtual architecure support them. This would hide chip complexity and changes from the compilers and interpreters.

Computing often recycles the old as new, and this holds true for strict virtual machines. Even the 360 name once used by IBM has recently been recycled in Yahoo!360 and Microsoft's Xbox 360. Perhaps strict virtual architectures will someday soon be pressed into use for multicore chips. These virtualities are of machinery. Is the principle extensible?

Programmers inhabit the next level up. Virtual programmers would be end users who could routinely put together sequences of high-level statements specific to their problems as users. They wouldn't be constrained by a set of buttons and templates designed for marketing reasons rather than user reasons by people living in a different world who didn't properly understand the problem area. These programmers could put basic operations together in their own sequences, subject to their own conditioning.

This virtual programming could be done through a command and scripting/macro interface. Thus another wheel would turn. ■

Neville Holmes *is an honorary research associate at the University of Tasmania's School of Computing. Contact him at neville.holmes@utas.edu.au. Details of citations in this essay, and links to further material, are at www.comp.utas.edu.au/users/nholmes/prfsn.*

Seapine Software™
changing the world of software development

Developers love creating code... Managers crave process control... Bridge the gap with Seapine CM.

Software development is a team effort with developers, testers, and management all working toward one goal – delivering the highest quality product on time.

Built on award-winning TestTrack Pro and Surround SCM, Seapine CM brings structure to source control and issue management, improving communication while accelerating product development.

Seapine CM helps your team...

Define custom change request workflows, putting you in control of who makes changes and who authorizes the closure of issues.

Associate source code changes with defects or change requests.

Gain a thorough understanding of how much work remains before project completion.

View complete audit trails of what changed, why, and by whom.

Understand how close you are to release—how many issues are open, how quickly are you closing them, how many are re-opened?

Successful team-based development requires the proper process supported by the right development tools. Tools that are flexible, easy to use, secure, and scalable–like Seapine CM.

Features:

Complete source code control with private workspaces, automatic merging, role-based security and more.

Comprehensive defect management—track change requests, bug reports, feature requests and more.

Fast and secure remote access to your source files and defects —work from anywhere.

IDE integration with JBuilder, Visual Studio, Dreamweaver, and other leading development tools.

Advanced branching, triggers, email notifications, fully configurable workflows, and customizable fields put you in complete control of your process.

Open interfaces—triggers, email notifications, SOAP support, XML data exchange, and ODBC clients.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Achieve major improvements in software development performance through better tool integration and process automation. Manage defects, development issues, and change requests with award-winning TestTrack Pro and gain complete control over your source code and change process with Surround SCM. Seapine's integrated change management tools are feature rich, highly scalable, Web enabled, and cross platform. Streamline your development process with Seapine CM and help your team deliver quality software products on time, every time.

Learn more about the Seapine CM suite at www.seapine.com or call 1-888-683-6456

Surround SCM Source Control

TestTrack Pro Defect Management

QA Wizard Automated Testing

software development 13th annual product excellence award