On the Marketing of Multicore

Matthew C. Bell   Patrick H. Madden
Santa Clara University   SUNY Binghamton
mbell@scu.edu   pmadden@acm.org

In the past few years, microprocessor architectures have undergone a fundamental change. Driven by a variety of factors, leading designs have transitioned from single monolithic processors to "multicore" configurations. In this paper, we survey prior work on parallel processing systems, and discuss the enthusiasm for multicore designs from a psychological perspective.

We argue that the semiconductor industry faces a difficult challenge. There is wide agreement that single-core processing rates have peaked, and that any further significant progress is unlikely. The shift towards parallel architectures is not necessarily a solution, however: parallel software and applications are fundamentally different from their serial counterparts, and the market for parallel computing has never been particularly large. Without a high volume and high profit product such as the consumer microprocessor, it is unclear where revenue will come from to drive forward with Moore's law scaling.

Categories and Subject Descriptors
J.6 [**Computer-Aided Engineering**]: CAD

General Terms
Algorithms

Keywords
Multicore, Wishful Thinking, Interconnect Optimization

1. INTRODUCTION

In this paper, we consider the commercial prospects for multicore microprocessors. We focus specifically on microprocessors in "personal computers;" sales of these chips have provided a great deal of the revenue required to support the semiconductor industry.

Under pressure to increase performance, designs have shifted from a single monolitic processor to multicore configurations. This shift began with the introduction of the IBM PowerPC Power4 in 2001; AMD, Sun, Intel, and Freescale have also announced or released multicore designs.

While the shift in some respects "solves" many problems (power being foremost), we argue that this comes with a heavy price. While many in the field are enthusiactic about multicore designs, we present a contrarian view. We argue that multicore designs will likely be a commercial failure for the consumer market.

To be clear: we restrict our focus to personal computers, where parallel computing has gained little traction despite many years of effort. Without widespread consumer adoption of parallel computing, the prospects for the semiconductor industry appear bleak. Moore's law[45] holds because *it is profitable* for semiconductor manufacturers; for fourty years, large numbers of consumers have paid high prices for the highest performance designs. The revenue from this market has enabled the research and development needed to move along the exponential growth path. If multicore microprocessors fail to impress consumers, sales will fall flat, and revenue will be reduced. A shortage of revenue will have a ripple effect throughout the semiconductor and EDA industries.

Our objective with this paper is to provide a counterbalance to the surge of research into "server farm on a chip" parallel computing[54], and to shed some light on to why there is such great enthusiasm for it. We also hope to spark debate on the appropriate direction of consumer microprocessor designs.

To provide context, in Section 2 we discuss prior attempts to commercialize parallel computing systems. Many different architectures, programming languages, compilers, and software libraries have been developed. Over the past fourty years, there have been many predictions of an impending golden age of parallel computing; this has yet to come to pass.

Despite the lack of commercial success in parallel computing, a number of major semiconductor manufacturerers have clearly stated that their long term plans are to move in this direction. Similarly, a great many researchers are actively working on the design of multicore architectures, and government funding[10] has been earmarked to support them. Dissenting opinions are scarce; Section 3 presents research performed by psychologists to explain why so many talented engineers and scientists may be "getting it wrong."

We conclude this paper with a short summary, and a call for viable suggestions as to what a consumer-grade "killer application" for massively parallel systems might be. Sales of vast numbers of laptop and desktop personal computers have provided much of the revenue for the semiconductor industry. The fundamental nature of the processors in these computers is changing; how this change will impact consumer demand has received very little public discussion.

2. A BRIEF HISTORY OF PARALLEL PROCESSING

Parallel processing has a long history[1]. Unfortunately, history has not been particularly kind to commercial ventures. While some

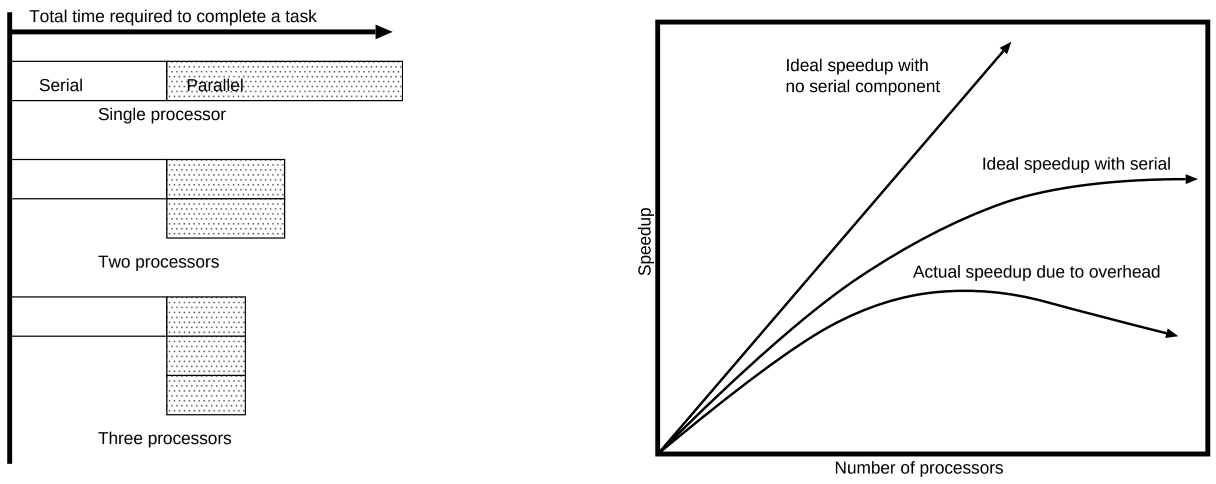
Total time required to complete a task
Serial
Parallel
Single processor
Two processors
Three processors
Ideal speedup with no serial component
Ideal speedup with serial
Actual speedup due to overhead
Speedup
Number of processors

Figure 1: Amdahl's law suggests that the speedup possible through parallel computation is asymptotic; while "parallel" portions have reduced run times, the serial portions are unaffected. Scaling beyond a few processors is rarely beneficial; communications overhead can in some cases result in a slowdown.

scientific applications can utilize parallelism (and achieve remarkable leaps in performance)[24], parallel processing has failed to reach a broad market.

A difficulty with parallel computing is that many problems have elements that are inherently serial. Amdahl's law[1] is well known; if a computing task has a serial component which takes S time, and a parallel component which takes P time, the run time with n processors is at best $S+P/n$. In practice, communications overhead between processors increases run times, and the amount of parallelism that can be exploited may not scale to large numbers of processors. This is illustrated in Figure 1.

If personal computers are to utilize multicore architectures effectively, some combination of the following must occur.

- Individual applications must utilize multiple processor cores. This is challenging, as parallel programming is known to be difficult. Further, many applications have relatively little inherent parallelism. Increasing problem sizes to better exploit parallelism (as done in [24]) is not practical for consumers who expect their applications to be interactive.

- Multiple compute intensive applications must run simultaneously. While there is some demand for simultaneous applications (e.g., an MP3 player in conjunction with a word processor), multitasking operating systems with single processor cores are frequently adequate. While a modern operating system may have many processes, the majority of these processes are idle at any given moment.

2.1 Parallel Hardware

On the surface, multi-processor systems are appealing on a number of levels. Algorithms that can be run in parallel can utilize multiple cores to reduce run times; if more than one compute-intensive application is running, system performance does not degrade. The design process for a multi-core system is also easier; small processor cores can be replicated, and arbitrarily large amounts of silicon area can be utilized. With a well designed on-chip interconnection network (for example [15, 40]), it is clearly possible to construct a "server farm on a chip[54]." In terms of the number of floating point operations possible per second, energy per operation, and a variety of other metrics, parallel computing has many advantages.

The advantages have been clear for many years, however–and many parallel systems have been constructed. Despite a great deal of effort, almost all commercial ventures into the construction of both small and large scale multi-processor systems have met little success. Perhaps most dramatic was the attempt by Thinking Machines in the mid 1980's; despite having an experienced architecture team with excellent credentials, as well as computer scientists and physicists of note, the company failed to become profitable. While the systems were functional, and could solve specific problems with unprecedented speed, their application was so narrow that very few systems were purchased.

The Thinking Machines systems employed thousands of commodity microprocessors; processors optimized for parallel computing have also been developed. As an example, thousands of Inmos Transputers were assembled into systems for parallel computation[51]. A consumer-level computer utilizing the Transputer was marketed by Atari[5]. Different configurations, numbers of processors, message passing schemes, and network topologies have all been explored (for example, [31, 30, 24, 6, 65]). Companies such as Meiko[53], Kendell Square Research, MasPar, NCUBE, Sequent, Parsytec[49], and Cray[2], have all implemented different levels of parallelism to achieve higher performance computing–and in each of these cases, the revenue from sales fell far short of corporate needs. Academic efforts to introduce parallel architectures have also had difficulty[62, 16].

For scientific or large server applications, parallel machines can come remarkably close to the theoretical limits[1], and the price and performance advantages are undeniable; to date, however, there has been little success in harnessing this power for "general purpose" computing. While parallel systems can be technologically impressive, they form only a small portion of the marketplace compared to the millions of consumer laptop and desktop machines.

Furht[22], in a 1994 paper, notes the following: *"a decade ago, university researchers were in love with parallel computers, and the US government amorously responded. Those were the days of glory, but times have changed: the market for massively parallel computers has collapsed, and many companies have gone out of business, but the researchers are still in love with parallel computing."*

We would argue that the lack of commercial success for parallel computing is not due to a lack of effort in hardware design.

2.2 Parallel Compilers

The difficulty faced by parallel architectures is also not simply an artifact of poor compilers, or the lack of effort on the part of software developers. Parallel-optimizing compilers have been in existance for many years[41, 19, 38, 57], and there have been numerous attempts to develop better mathematical frameworks. As an example, one might consider the programming language *occam*, developed in the early 1990s–carefully designed, flexible, portable, and essentially non-existant in modern computing.

Some compilers have been specially designed for specific parallel machines. The J-Machine[62, 16] supported fine-grained parallel programs (one might note similarities between the designs of the J-Machine and more recent "network on chip" proposals[15]).

Over the past thirty years, FORTRAN has had many variants which have supported parallel architectures[36, 25, 52, 28]. The level of FORTRAN support is indicative of the demand for parallel resources amongst the scientific community.

Functional languages have also supported parallel architecture for decades. Compilers for Prolog[34, 50, 23], Lisp[35], and other similar languages[9, 12, 20, 66] have been implemented.

Main-stream languages such as C, C++, and Java, also have robust support for parallel hardware[26, 27, 3, 71, 17] Versions of Ada, which was heavily supported by federal funding, also have support for parallel architectures[14]. New programming languages and environments are also being developed (e.g., Chapel[11]).

Platform independent libraries such as OpenMP[55] and MPI[7] have been developed, greatly simplifying message passing between processors. We would note that these have been used by design automation tools developers (for example, in the *feng shui placement* tool[32]). EDA researchers are certainly familiar with parallel programming resources, and have applied them when appropriate.

2.3 Multi-Tasking

Some might argue that multi-tasking operating systems are a natural fit for multicore processors. Multi-tasking has been commonplace for years, however, with most systems working quite well with a single CPU.

A typical consumer might have dozens of compute tasks active at any point in time – but the vast majority of these spend a great deal of time in idle mode. One might have a task which checks for email or incoming instant messages; a dedicated processor would allow these checks to happen millions of times per second, but this is far in excess of what a user requires.

2.4 The Commercial Prospects

In the mid 1980's, academic and industry researchers had great confidence in the commercial prospects of parallel computing; history shows that this confidence was misplaced. From the earliest days, parallel architectures have been successful with some scientific applications, but there is little success elsewhere.

Despite this, massively parallel computing has returned as a primary target of government funding[10]. At one point, federal agencies were "in love" with parallel computers[22]; it appears that the love has blossomed again.

The resurgence is not due to any significant advance that has made parallel resources more palatable, or easier to use. Software development for parallel machines remains difficult, and improvements are becoming asymptotic[37]. We would argue that the challenges and pitfalls encountered by the prior generation of researchers and developers are likely to be encountered again. From a theoretic perspective, a fundamental change in processor architectures was inevitable[44]; this change was delayed for as long as possible. While parallel computing has apparently "arrived" for the consumer market, we wish to stress that this should be viewed as a *last resort*.

3. UNDERSTANDING THE ENTHUSIASM

Considering the prior commercial failures of consumer parallel processing, the shift to multicore is being greeted with a surprising amount of enthusiasm. In some respects, the battle for the fastest clock rate has evolved into a battle to have the most processor cores. Four core systems are anticipated in 2007, with eight cores (a "peak" number suggested in [4]) following shortly thereafter.

We would argue that the enthusiasm is misguided, and that the situation has been misjudged by the community. Obviously, time will tell if we are correct in our assertion. We would note that in the past, a general consensus of the engineering and research community has not always been correct (e.g., fifth-generation programming languages, or the dot-com investment craze).

In this section, we discuss reasons why there is enthusiasm, drawing on research performed by psychologists over a number of years. An overview of research in this area can be found in a text by Cialdini[13].

3.1 Decision Making Errors

Irrational or erroneous decisions can occur for a variety of reasons. Some of the more common reasons, related to the problem facing the wide-scale implementation of dual-core technology, are described briefly below. The list is not exhaustive; rather it is intended to illustrate some of the primary problems facing the integrated circuit design community.

3.1.1 Mental sets

People frequently find a solution to a problem and persist in applying the solution to every new problem. Luchins[43], in a classic experiment, asked subjects to work a series of problems. The first four problems required the same strategy for developing a solution. The fifth problem, however, had two paths to the solution: one that required the cumbersome but proven strategy employed on the previous problems and one significantly simpler solution. Subjects persisted in using the more cumbersome strategy. Additionally, when given a sixth problem for which the proven strategy did not work, subjects continued to attempt to solve the problem using the formerly functional strategy. The tendency to continue with a rigid problem-solving approach rut is a common problem. Smith[61] and Wiley[70] may explain why experts fail in problem-solving efforts[39].

With respect to the design of microprocessors, the single core architecture had a remarkable series of successes over several decades. The mental set issue manifests itself in an initial reluctance to switch toward a multicore architecture. The adoption of multicore architecutres is a different mental set; the solution strategy has had success in supercomputing applications, and there is now an attempt to apply it to the consumer market.

3.1.2 Risky decision making

Risky decision making occurs when people make choices under uncertain conditions. Gamblers, for example, frequently behave in ways that are inconsistent with expected value[60], and, in fact, often engage in behaviors where the expected value is negative (i.e., they should lose money). Research suggests that when people make decisions that violate expected value that they are relying on subjective utility[21], where an individual evaluates the personal worth of the outcome. Interestingly, people quite often make poor predictions (i.e., underestimate) about the subjective value of the outcome[42].

With respect to design teams and researchers, it is interesting to note the relative risks of pursuing multicore architectures. For circuit designers tasked with increasing performance, the "success" of a project is measured by the number of instructions per second, but not on the eventual sales. To remain employed, the best strategy may be to target a design that will in fact have little commercial appeal. Similarly, researchers seeking funding have good prospects with proposals targetting parallel computation[22, 10]. A lack of commercial impact for the research does not seem to degrade the chances of securing funding.

3.1.3 Ignoring base rates

People often ignore the actual rates that events occur. For example, Weinstein[68, 69] reported that people underestimate the risks of their own negative health-related behaviors (e.g., smoking). Additionally, the more variables coming into play and the further in the future the event is scheduled to occur, the more likely it is that people will make bad predictions[47].

There are well documented cases where parallel computation has been successful. We would argue that it is likely that the successes are overestimated, while failures are downplayed.

3.1.4 Heuristics

People use heuristics to judging the probability that an event will occur. There are a number of different ways people make estimates, all of which are likely to result in poor decisions being made. For example, people often base their estimates of the likelihood of an event based on how easy it is for them to remember examples (availability heuristic), or how closely the example matches a prototypical example (representativeness heuristic).

3.1.5 Gambler's Fallacy

People believe that the odds of a chance event increase because the event has not recently occurred[63, 67]. In fact, even when they are given explicit instructions about the fallacy, they will continue to bet and invoke the gambler's fallacy[8].

In discussions with colleagues, we have found that some believe that parallel computing will be successful *because it has not been so previously.* The repeated predictions of commercial success, coupled with the obvious lack thereof, seem strengthened the belief.

3.1.6 Overestimating the improbable

People will greatly overestimate the odds of infrequent events that are dramatic and vivid if, for example, they receive substantial media coverage[59].

3.1.7 Confirmation bias

We have the tendency to seek and use information that supports our beliefs and decisions[46] and to verify our ideas more eagerly and fail to seek or believe refuting evidence[33, 58].

3.2 Groupthink

Groupthink is a phenomena that occurs when members of a group emphasize concurrence at the cost of critical thinking in decision making. Janis [29] (and also Eaton, [18]) suggested that at least some subset of the following antecedent conditions are necessary for groupthink to occur were (a) high group cohesiveness, (b) insulation of the group, (c) lack of methodical procedures for search and appraisal, (d) directive leadership, and (e) high levels of stress with low chances of obtaining better solutions than the ones favored by the leadership. When groups engage in groupthink, individual members of the group abandon critical judgment, the group begins to censor dissenting views and the pressure to conform increases. When the group's view is challenged by an outsider, people engaged in groupthink will tend to over-simplify things as being an 'us versus them' situation. During groupthink, members will overestimate the ingroup's unanimity and will view the outgroup as the enemy. Additionally, groupthink promotes incomplete gathering of information (e.g., confirmation bias)[56]. Finally decisions made as a group typically result in polarization of the group's views. In fact, group discussion will strengthen the dominant view, shifting it toward a more extreme, riskier decision[48, 64].

In particular, condition (e) seems to capture the current state of the industry well. Design teams are under intense pressure to produce faster microprocessors; Intel had a pair of high-profile design cancellations while AMD produced a successful multicore design. While the circuit design and EDA communities may pride itself on good engineering and scientific practices, the environment is ripe for groupthink.

4. CONCLUSION

From 1965 to 2001, the semiconductor industry used a "single-core" architectural model, in which microprocessors consumed all available transistors. There was great consumer demand at each step along the way, and all available processing power was easily harnessed. Binary compatibility of processor families allowed performance gains to be obvious and immediate.

From 2001 to current designs, there has been a shift to multicore architectures. For this approach to be successful, programmers will need to rewrite many applications, new software must be developed, and many existing application programs will need to be abandoned. The throughput of individual cores will remain static, or perhaps even degrade–for consumers to see any benefit at all, multiple compute-intensive applications must run simultaneously. To sustain Moore's law, the number of cores must also grow exponentially; with each generation, it will become increasingly difficult to find ways to utilize the available computing resources.

We would summarize the change as follows.

- The industry has elected to **stop developing higher performance single core microprocessors.** Power is a primary limiting factor, but there are a host of other good reasons for this decision. A high volume, high price, and high profit margin product, which has proven to be in great demand by consumers, is being abandoned. There is little doubt that if a single core microprocessor were able to offer 2*x* the performance of current designs, consumers would readily pay far more than 2*x* the price.

- As a replacement for the single core microprocessor, the industry is offering **flat performance for each core, but exponentially increasing numbers of cores.** This avenue has been chosen despite the fact that similar ideas have been explored for four decades, without significant commercial success.

- It is assumed that consumers will adopt the new architectures because **new, but as of yet unknown applications** will be developed. Existing applications may be **migrated to parallel implementations**, despite the fact that most users are reluctant to switch software. Simply needing to recompile software hampered the introduction of some chips (e.g., the

DEC Alpha); yet it is assumed that a much more difficult hurdle will be passed easily.

We anticipate that many will disagree with our assertion that consumers will not adopt parallel processing. As was noted[22], many staunchly believe in the promise of consumer parallel computing, despite all evidence to the contrary. With a brief consideration of the history of parallel processing, one finds the unbridled optimism of researchers, engineering teams, and investors, juxtaposed against stunning, relentless, commercial failure. Multicore processors are certainly useful for high-capacity servers and supercomputers–but this is only a small portion of the market.

The semiconductor industry now faces a difficult challenge. There is no guarantee that multicore designs will be embraced, and good reason to expect that they will fail commercially. A primary source of revenue will no longer be at the leading edge of Moore's law. While a great deal of research focus has been on technical barriers to device scaling, the most significant barrier may be economic.

Our objective with this paper is to spark discussion regarding the shift towards multicore architectures. It is a simple fact that the shift has occurred – but it is open to debate as to if this is a wise decision.

We also wish to spur discussion, in specific terms, as to what might constitute a "killer application" for the new generation of processors. While it is easy to hope that such an application will be created, the chances of this occuring are greater if it is communicated to the software community that there is an imperative need for one.

5. REFERENCES

[1] G. M. Amdahl. Validity of the single-processor approach to achieveing large scale computing capabilities. In *Proc. AFIPS Conference*, pages 483–485, 1967.

[2] E. Anderson, J. Brooks, C. Grassl, and S. Scott. Performance of the CRAY T3E multiprocessor. In *Proc. Conference on High Performance Networking and Computing*, pages 1–17, 1997.

[3] Eshrat Arjomandi, Ivan Kalas, and William O'Farrell. Concurrency abstractions in a c++ class library. In *CASCON '93: Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research*, pages 919–932. IBM Press, 1993.

[4] A. Aston. BusinessWeek online article: More life for moore's law (http://www.businessweek.com/magazine/content/05_25/b3938629.htm, 2005.

[5] Atari. The Atari transputer (http://www.atarimuseum.com/computers/16bits/transputer.html), 1988.

[6] M. L. Barton and G. R. Withers. Computing performance as a function of the speed, quantity, and cost of the processors. In *Proc. ACM/IEEE Conference on Supercomputing*, pages 759–764, 1989.

[7] Ayon Basumallik and Rudolf Eigenmann. Towards automatic translation of openmp to mpi. In *ICS '05: Proceedings of the 19th annual international conference on Supercomputing*, pages 189–198, New York, NY, USA, 2005. ACM Press.

[8] D. M. Boynton. Superstitious responding and frequency matching in the positive bias and gamber's fallacy effects. *Organizational Behavior & Human Decision Processes*, 91:119–127, 2003.

[9] Timothy A. Budd. An apl compiler for a vector processor. *ACM Trans. Program. Lang. Syst.*, 6(3):297–313, 1984.

[10] G. W. Bush. State of the union address, 2006.

[11] D. Callahan, B. L. Chamberlain, and H. P. Zima. The cascade high productivity language. In *9th International Workshop on High-Level Parallel Progamming Models and Supportive Environments (HIPS 2004)*, pages 52–60, 2004.

[12] M. Chen, Y. Choo, and J. Li. Crystal: from functional description to effi cient parallel code. In *Proceedings of the third conference on Hypercube concurrent computers and applications*, pages 417–433, New York, NY, USA, 1988. ACM Press.

[13] R. B. Cialdini. *Influence: The Psychology of Persuasion*. Quill, 1993.

[14] B. Cockerham. Parallel compilation of ada units. In *TRI-Ada '88: Proceedings of the conference on TRI-Ada '88*, pages 147–164, New York, NY, USA, 1988. ACM Press.

[15] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. Design Automation Conf*, pages 684–689, 2001.

[16] William J. Dally, Andrew Chien, Stuart Fiske, Waldemar Horwat, Richard Lethin, Michael Noakes, Peter Nuth, Ellen Spertus, Deborah Wallach, D. Scott Wills, Andrew Chang, and John Keen. Retrospective: the j-machine. In *ISCA '98: 25 years of the international symposia on Computer architecture (selected papers)*, pages 54–58, New York, NY, USA, 1998. ACM Press.

[17] Timothy E. Denehy and Chang-Hyun Jo. Parallel-c++ for the java virtual machine. In *SAC '00: Proceedings of the 2000 ACM symposium on Applied computing*, pages 843–848, New York, NY, USA, 2000. ACM Press.

[18] J. Eaton. Management communications: The threat of groupthink. *Corporate Communications*, 6:183–192, 2001.

[19] Clarence A. Ellis. Parallel compiling techniques. In *Proceedings of the 1971 26th annual conference*, pages 508–519, New York, NY, USA, 1971. ACM Press.

[20] Marc Feeley and James S. Miller. A parallel virtual machine for effi cient scheme compilation. In *LFP '90: Proceedings of the 1990 ACM conference on LISP and functional programming*, pages 119–130, New York, NY, USA, 1990. ACM Press.

[21] B. Fischhoff. Judgment and decision making. In R. J. Sternberg and E. E. Smith, editors, *The Psychology of Human Thought*. Cambridge University Press, 1999.

[22] B. Furht. Parallel computing: Glory and collapse. *IEEE Computer*, 27(11):74–75, 1994.

[23] Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, and Manuel V. Hermenegildo. Parallel execution of prolog programs: a survey. *ACM Trans. Program. Lang. Syst.*, 23(4):472–602, 2001.

[24] J. L. Gustafson. Reevaluating Amdahl's law. *Communications of the ACM*, 31(3):532–533, 1988.

[25] M. D. Guzzi, J. P. Hoeflinger, D. A. Padua, and D. H. Lawrie. Cedar fortran and other vector and parallel fortran dialects. In *Supercomputing '88: Proceedings of the 1988 ACM/IEEE conference on Supercomputing*, pages 114–121, Los Alamitos, CA, USA, 1988. IEEE Computer Society Press.

[26] Philip J. Hatcher, Anthony J. Lapadula, Robert R. Jones, Michael J. Quinn, and Ray J. Anderson. A production-quality c* compiler for hypercube multicomputers. In *PPOPP '91: Proceedings of the third ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 73–82, New York, NY, USA, 1991. ACM Press.

[27] Philip J. Hatcher, Michael J. Quinn, Ray J. Anderson, Anthony J. Lapadula, Bradley K. Seevers, and Andrew F. Bennett. Architecture-independent scientifi c programming in data parallel c: three case studies. In *Supercomputing '91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 208–217, New York, NY, USA, 1991. ACM Press.

[28] Seema Hiranandani, Ken Kennedy, and Chau-Wen Tseng. Evaluation of compiler optimizations for fortran d on mimd distributed memory machines. In *ICS '92: Proceedings of the 6th international conference on Supercomputing*, pages 1–14, New York, NY, USA, 1992. ACM Press.

[29] I. L. Janis. *Victims of Groupthink*. Houghton Mifflin, 1972.

[30] A. K. Jones and P. Schwarz. Experience using multiprocessor systems – a status report. *Computing Surveys*, 12(2):121–165, 1980.

[31] P. H. Enslow Jr. Multiprocessor organization – a survey. *Computing Surveys*, 9(1):103–129, 1977.

[32] F. Khundakjie, N. B. Abu-Ghazaleh, M. C. Yildiz, and P. H. Madden. Parallel VLSI standard cell placement on a cluster of workstations. In *Proc. IEEE Clusters*, pages 529–553, 2001.

[33] J. Klayman and Y.-W. Ha. Confi rmation, disconfi rmation, and information in hypothesis testing. *Psychological Review*, 94:211–228, 1987.

[34] Alexis Koster. Compiling prolog programs for parallel execution on a cellular machine. In *ACM 84: Proceedings of the 1984 annual conference of the ACM on The fifth generation challenge*, pages 167–178, New York, NY, USA, 1984. ACM Press.

[35] D. A. Kranz, Jr. R. H. Halstead, and E. Mohr. Mul-t: a high-performance parallel lisp. In *PLDI '89: Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation*, pages 81–90, New York, NY, USA, 1989. ACM Press.

[36] Howard E. Krohn. A parallel approach to code generation for fortran like compilers. In *Proceedings of the conference on Programming languages and compilers for parallel and vector machines*, pages 146–152, 1975.

[37] Diogenes Lafong. Private communication, 2006.

[38] Leslie Lamport. On programming parallel computers. In *Proceedings of the conference on Programming languages and compilers for parallel and vector machines*, pages 25–33, 1975.

[39] J. P. Leighton and R. J. Sternberg. Reasoning and problem solving. In A. F. Healy and R. W. Proctor, editors, *Handbook of Psychology, Vol. 4: Experimental Psychology*. Wiley, 2003.

[40] T. Lin and L. T. Pileggi. Throughput-driven IC communication fabric synthesis. In *Proc. Int. Conf. on Computer Aided Design*, pages 274–279, 2002.

[41] Neil Lincoln. Parallel programming techniques for compilers. *SIGPLAN Not.*, 5(10):18–31, 1970.

[42] G. Loewenstein and D. Schkade. Wouldn't it be nice? predicting future feelings. In D. Kahneman, E. Diener, and N. Schwartz, editors, *Well-Being: The Foundations of Hedonic Psychology*. Sage, 1999.

[43] A. S. Luchins. Mechanization in problem solving. *Psychological Monographs*, 54(248), 1942.

[44] P. H. Madden. Supersized VLSI: A receipe for disaster. In *Electronic Design*

Processes, 2005.

[45] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 38(8):114–117, April 1965.

[46] R. S. Nickerson. Confi rmation bias: A ubiquitous phenomenon in many guises. *Review of General Psychology*, 2:175–220, 1998.

[47] R. A. Olsen. Desirability bias among professional investment managers: Some evidence from experts. *Journal of Behavioral Decision Making*, 10:65–72, 1997.

[48] D. G. Pruitt. Choice shifts in group discussion: An introductory review. *Journal of Personality and Social Psychology*, 20:339–360, 1971.

[49] G. Ramanathan and J. Oren. Survey of commercial parallel machines. *ACM SIGARCH Computer Architecture News*, 21(3):13–33, 1993.

[50] B. Ramkumar and L. V. Kale. A chare kernel implementation of a parallel prolog compiler. In *PPOPP '90: Proceedings of the second ACM SIGPLAN symposium on Principles &amp; practice of parallel programming*, pages 99–108, New York, NY, USA, 1990. ACM Press.

[51] F. Roch-Siebert and G. Villard. PAC: First experiements on a 128 transputers meganode. In *Proc. International Symposium on Symbolic and Algebraic Computation*, pages 343–351, 1991.

[52] Roland Rohl and Marco Annaratone. Parallelization of fortran code on distributed-memory parallel processors. In *ICS '90: Proceedings of the 4th international conference on Supercomputing*, pages 342–353, New York, NY, USA, 1990. ACM Press.

[53] D. Roweth. The Meiko CS-2 system architecture. In *Proc. ACM Symp. on Parallel Algorithms and Architectures*, page 213, 1993.

[54] M. Santarini. EE Times article: Cal Berkeley dean predicts server-farm-on-a-chip (http://www.eedesign.com/article/showarticle.jhtml?articleid=51000480, 2004.

[55] Mitsuhisa Sato. Openmp: parallel programming api for shared memory multiprocessors and on-chip multiprocessors. In *ISSS '02: Proceedings of the 15th international symposium on System Synthesis*, pages 109–111, New York, NY, USA, 2002. ACM Press.

[56] S. Schulz-Hardt, D. Frey, C. Luthgens, and S. Moscovici. Biased information search in group decision making. *Journal of Personality and Social Psychology*, pages 655–669, 2000.

[57] David B. Skillicorn and Domenico Talia. Models and languages for parallel computation. *ACM Comput. Surv.*, 30(2):123–169, 1998.

[58] R. B. Skov and S. J. Sherman. Information-gathering processes: Diagnosticity, hypothesis-confi rmation strategies, and perceived hypothesis confi rmation. *Journal of Experimental Social Psychology*, 22:93–121, 1986.

[59] P. Slovic, B. Fischhoff, and S. Lichtenstein. Facts versus fears: Understanding perceived risk. In D. Kahneman, P. Slovic, and A. Tversky, editors, *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, 1982.

[60] P. Slovic, S. Lichtenstein, and B. Fischhoff. Decision making. In R. C. Atkinson, R. J. Herrnstein, G. Lindzey, and R. D. Luce, editors, *Stevens' Handbook of Experimental Psychology (Vol. 2)*. Wiley, 1998.

[61] S. M. Smith. Getting into and out of mental ruts: A theory of fi xation, incubation and insight. In R. J. Sternberg and J. E. Davidson, editors, *The Nature of Insight*, pages 229–251. MIT Press, 1995.

[62] Ellen Spertus, Seth Copen Goldstein, Klaus Erik Schauser, Thorsten von Eicken, David E. Culler, and William J. Dally. Evaluation of mechanisms for fi ne-grained parallel programs in the j-machine and the cm-5. In *ISCA '93: Proceedings of the 20th annual international symposium on Computer architecture*, pages 302–313, New York, NY, USA, 1993. ACM Press.

[63] K. E. Stanovich. The fundamental computational biases of human cognition: Heuristics that (sometimes) impair decision making and problem solving. In J. E. Davidson and R. J. Sternberg, editors, *The Psychology of Problem Solving*. Cambridge University Press, 2003.

[64] R. S. Tindale, T. Kameda, and V. B. Hinsz. Group decision making. In M. A. Hogg and J. Cooper, editors, *The Sage Handbook of Social Psychology*. Sage, 2003.

[65] P. T. Tosic. A perspective on the future of massively parallel computing: Fine-grain vs. coarse-grain parallel models, comparison and contrast. In *Proc. Conference on Computing Frontiers*, pages 488–502, 2004.

[66] P. W. Trinder, H.-W. Loidl, and R. F. Pointon. Parallel and distributed haskells. *J. Funct. Program.*, 12(5):469–510, 2002.

[67] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. In D. Kahenman, P. Slovic, and A. Tversky, editors, *Judgment Under Uncertainty: Heuristics and Biases*. Cambridge University Press, 1982.

[68] N. D. Weinstein. Why it won't happen to me: Perceptions of risk factors and susceptibility. *Health Psychology*, 3:431–458, 1984.

[69] N. D. Weinstein and W. M. Klein. Resistance of personal risk perceptions to debiasing interventions. *Health Psychology*, 14:132–140, 1995.

[70] J. Wiley. Expertise as mental set: the effects of domain knowledge in creative problem solving. *Memory and Cognition*, 26:716–730, 1998.

[71] Yingchun Zhu and Laurie J. Hendren. Communication optimizations for parallel c programs. In *PLDI '98: Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation*, pages 199–211, New York, NY, USA, 1998. ACM Press.