

The Case for a Wireless Platform

Philip Machanick
School of IT and Electrical Engineering, University of Queensland
Brisbane, QLD 4072
Australia
philip@itee.uq.edu.au

Abstract

Wireless computing and more general wireless devices are implemented with a wide range of technologies. As can be expected from an emerging niche, there are many problems in finding good compromises between standardizing and allowing flexibility to experiment. The compelling ideas of ubiquitous and pervasive computing make a case for standards to support interoperability. At the same time, the relative immaturity of the area makes a case for flexibility. This paper examines architectural issues in defining a wireless platform, taking into account lessons of other successful architectures. The proposed starting point is to maintain a clean distinction between communications and application aspects of architecture. Communications, because of the inherent flexibility of the wireless medium, should be kept as open and flexible as possible, while flexibility of an application architecture is less critical, given that small devices implementing a specific application are becoming increasingly feasible.

1 Introduction

Low-energy¹ computing is a growing area of research [1, 2, 3, 4, 5, 6]. Mobile devices need low-energy designs, but low energy also benefits high-performance computing because high power consumption limits scalability. Low energy use is increasingly being treated as a primary rather than secondary design concern. While addressing low energy as a concern in traditional computing is likely to be an increasing trend, the potential for mobile devices is increased by improved power management, lower-energy devices, and improved battery technology.

Mobile devices historically have been sharply divided into full-scale computers (notebooks and PDAs) and special-purpose devices (mobile phones, cameras, MP3 music players). As the cost of processor and memory technology drops, some convergence of capabilities should be expected. More significantly, improvements in speed of wireless communication holds the promise that special-purpose devices can be integrated into previous unanticipated combinations.

This paper focuses on broad issues in tapping into this trend of increased utility of low-energy computing from the perspective of specialized mobile devices. However, the potential for convergence with conventional computing suggests that such a perspective should intersect with conventional computing concerns.

Much of the work on integrating such devices into wider services has looked at them from the perspectives of networking and usability – pervasive [7] or ubiquitous computing [8] are common keywords.

One of the key issues inhibiting realizing the vision of ubiquitous or pervasive computing is a lack of architectural standards. There is a wider variety of processors in mobile devices than in common desktop devices, there are no established standards for comparing performance, and special-purpose computation, such as in digital signal processors (DSP) plays a significant role. Added to this the fact that network protocols are able to evolve more rapidly than with a wired network (if only because there are no fixed cables to upgrade), and the overall picture is one of a confusion of choices, rather than a seamless platform.

Given that wireless operation introduces potential for highly diverse operation (aside from a mutable physical layer, its inherent mobility increases the potential range of applications [9]), it is useful to take a wireless platform as a basis for investigating general low-energy computing issues. Drawing a dividing line between application and communication issues is a useful abstraction (which has led to the idea of coordination languages in distributed

¹“Energy” and “power” are loosely referred to as the same thing but since power is the rate of energy usage, it does not capture the property of most significance to many designs, the peak energy usage.

systems [10]). However, given that mobility, wireless operation and low-energy design are often common concerns, it is useful to group these concerns together in one overall platform.

In the field of automotive electronics, a convincing case has been made for platform-based design, as opposed to existing ad hoc approaches. Many of the same arguments apply here – particularly the need to exploit design reuse and the separation of concerns (communication versus behaviour) [11].

The approach taken in this paper is to bring together ideas from a number of sources, to propose a general direction for future research. The intent is to position wireless computing in the mainstream, where there are well-defined architectures including widely-accepted performance benchmarks, and interoperability standards.

1.1 The Challenge

Some may wonder why it is useful to define an architecture for wireless systems. Current designs illustrate the value of the lack of an architecture: diversity of implementation techniques, rapid evolution of network protocols and a wide variety of application areas suggest that the area of wireless devices is one in which rapid advance is occurring.

However, this rapid advance is not driven by a consistent model, which results in patchy advances. Much of the improvement is in technologies, not in applications. The ideas behind pervasive and ubiquitous computing are not new. But only after a decade or so of active work has the notion of a consistent application model started to emerge [12].

Some of the reason for this lag has been the fact that small mobile devices have in the past lacked the processing power to do more than a limited range of tasks. A mobile phone, for example, in the early stages of digital phone technology, could be expected to maintain a list of phone numbers and ring tones, and not much more than that. As the cost of processor logic and memory reaches a level where more sophisticated applications become possible, an application architecture becomes an issue.

However, as long as devices are limited to running their own special-purpose software (possibly with options of augmentation and upgrading), the case for a general architecture is limited. Once devices become capable of running more than a minimal set of tasks, the possibility is opened up of more open-ended interactions with their environment – including other mobile devices.

The challenge as these options open up is to maintain as much of the design flexibility of traditional mobile devices as possible, while defining standard platforms as a baseline for interoperability and performance analysis.

Given that others are already working on application models, it is useful to start paying attention to architecture issues – processor design, memory systems and the hardware-software interface. Further, the application model is not as critical in this style of architecture as in traditional systems: if a specific device is limited in the range of applications it runs, as long as communication is possible, it can exchange information with other devices providing missing functionality.

1.2 Architecture Issues

To understand architecture issues in an emerging area, it is instructive to look back at the early history of the current mainstream. Before the IBM 360 series introduced the idea of a consistent architecture, computer manufacturers generally designed completely different computers to address different niches, with different instruction set architectures, application models and operating systems.

While IBM lost some design flexibility by designing a single architecture to cover all niches, the fact that their customers had a clear upgrade path was a major competitive advantage.

More significantly than bettering the initial competition, the IBM 360 architecture persisted over more than 20 years. It is useful to consider a few reasons for this persistent success [13]. First, the design consciously took into account the needs of not only scaling up, but also down: the lowest-end design had to be capable of running the same software as the highest-end design. Second, there was a careful distinction between aspects of the design which had to be tightly specified and those which did not.

These two principles on their own are interesting enough to bear further analysis.

“Scalability” is often considered only in terms of being able to create really large systems. However, the history of exotic supercomputer manufacturers suggests that the inability to operate at a variety of scales including very small is risky. Any wireless platform proposal which is going to see wide use would do well to include the possibility of very small-scale devices and combinations of such devices, because that is where the mass market is likely to be. If such a design can leverage the mass market to make developing large-scale applications cheaper, developers and users of large-scale applications can benefit from economies of scale beyond their own niche.

The distinction between aspects of a design which need to be precise and those which should not be over-specified is an important engineering principle in general. In a new niche, especially one where imprecision has historically been an advantage, care needs to be taken in delineating the imprecise-precise boundary. As the IBM 360 experience has shown, careful choice in this area can make for an architecture with a very long life.

While many other architecture issues can arise, the two guiding principles of scalability up and down, and careful choice of what should be precisely specified and what should not are a sufficient starting point for exploring a fuller range of issues.

1.3 Remainder of Paper

The remainder of this paper is structured as follows. Section 2 outlines some established issues and technologies as a starting point. Section 3 revisits problems and challenges extracted from previous discussion, focusing on possible solutions. The paper concludes with a summary of the major issues, and proposes a way ahead.

2 Background

Mobile devices (with less computation generality than PDAs or full computers) have traditionally been designed around special-purpose processors and operating systems. In the “smart phone” sector, for example, Symbian dominated the early operating system market with about 70% of units shipped in the first quarter of 2003 [14]. Network protocols have also varied from those designed for desktop systems, though there is an increasing convergence in the networking world to support interoperability or at least data synchronization.

This section examines some of the key technologies in this market, with a view to identifying aspects which need attention, if a common platform is to emerge. Issues which have been considered conventional in computer architecture research are examined, to see how they still apply in this area.

Processor design in conventional computers has gone through a number of generations, with relatively clear lessons for designers who wish to have a long-term influence. Mistakes are often repeated when a new niche opens up because the short-term design trade-offs which apply at the early stages of a new niche are often the same as those when a previous niche opened. For example, mainframe designers were surprised at the dramatic performance improvement Seymour Cray was able to achieve in the Control Data 6600 by simplifying the instruction set design [15]. Yet early microprocessor designers ignored this lesson, and instead, focused on encoding instructions to reduce memory usage – borrowing somewhat from the heritage of complex instruction sets of minicomputer designs, like the VAX.

As mobile devices start to converge with conventional computing, processor design choices in this market are worth examining for a reasonable balance between short-term trade-offs and long-term viability.

Operating systems too represent an area where lessons are forgotten – or deemed irrelevant in the face of short-term considerations. For example, when personal computers first appeared, long-understood concepts like virtual memory and preemptive scheduling were deemed irrelevant, and had to be retrofitted to the two most popular systems (Windows and Macintosh) at great expense both to vendors and users.

In the networking field, lessons in opening up new niches also exist. One of the most popular errors is attempting to establish a proprietary protocol when an open standard already has headway. This mistake has been repeated in several key niches, including mainframes and personal computers.

The remainder of this section examines these issues further, starting with processors, followed by operating systems, and, finally, networks. The section concludes with a summary.

2.1 Processors

Processors in mobile devices fall into three categories: specialized designs, mainly digital signal processors (DSPs), conventional designs targeted specifically to low-energy applications, and low-energy variants of more widely-used processors.

In a conventional computer, it could be an option, should there be a case for operations to support some peripheral device, to move that functionality to a device controller. In small mobile devices, cost, power budget and size drive minimizing component count, so there is a stronger temptation to design special-purpose instructions into a processor. This temptation has been realized in the form of specialized processors with DSP instruction sets, and extended instruction sets such as the AltiVec extensions of the PowerPC.

The remainder of this subsection examines the differences between the design philosophies of DSP instruction sets, extended instruction sets and conventional RISC architectures. First, DSP design issues are briefly considered, followed by some issues in choosing between low-energy conventional processors. Challenges in combining requirements of conventional designs and DSPs lead to issues for hybrid designs.

2.1.1 DSP Designs

Much work on DSP designs has focused on improving the speed of kernels (fragments of code contrived to approximate specific conditions) or fragments of real code, a strategy abandoned some time ago in conventional computer architecture. The strongest argument against benchmarking using kernels (Linpack, Livermore Loops), fragments of real code or even short combinations of instructions (multiply-accumulate or MAC is a common basis for comparison of DSP designs) is that they are not real programs, and carry the risk of encouraging instruction-set tweaks which no compiler can use in practice [16]. Even if they do represent performance of some aspect of a program, Amdahl's Law says that focusing effort on improving one aspect of performance without improving everything else has limited value.

It would be easy to criticise DSP designers for ignoring this key lesson of conventional architecture, but care needs to be taken to consider cases where Amdahl's Law does not apply. In a case where a specific operation is time-critical but others are not, it does make sense to focus on that operation. For example, in a DSP application in a mobile phone, voice encoding is time-critical and can justify a special-purpose design [17] – even if the effect might be slower processing averaged over all operations.

Even given that DSP applications may include cases where Amdahl's Law does not apply in the usual way, DSP designs are susceptible to some of the improvements seen in microprocessor design in recent decades. The RISC movement has attacked specific instruction set architecture (ISA) features as inhibiting efficient implementation: multiple addressing modes, variable instruction sizes and rarely-used instructions which are hard to implement.

On the one hand, common DSP designs appear to break some of the rules, in that they have complex addressing modes and a wide array of different instructions. On the other, the inherent flexibility of a market where programs are small-scale and special-purpose has allowed some recent designs to incorporate features ahead of conventional architectures, such as a combination of predicated instructions and tags explicitly specifying parallelism [18].

A key issue to consider with DSP designs is whether the specialized instructions and addressing modes are critical to the real-time operations typically performed on DSPs. To compare them against conventional architectures on the basis of an Amdahl's Law-style performance analysis could miss the important difference in performance criteria between real-time and "conventional" applications. A real-time application (especially if it's hard real time – the time constraint has to be met, otherwise the system fails) has to be measured against worst-case scenarios, a very different performance constraint to other applications, where average performance is often an adequate measure [19].

A big challenge for computer architecture researchers used to the world of SPEC benchmarks is how to evaluate performance of processors aimed at meeting hard real-time requirements. Unlike with conventional system performance measures, simply achieving a faster overall runtime is not sufficient. Standard performance measures should take into account response time for real-time tasks. Scaling performance up should include not only faster overall run time, but being able to solve more ambitious real-time tasks. (For example, implementing higher-quality audio codecs, or implementing a higher quality of video.)

Emerging standards for embedded systems benchmarks, such as those of the Embedded Microprocessor Benchmark Consortium² must address these concerns.

2.1.2 Low-Energy Conventional Designs

Desktop and server processors (Intel and clones versus the rest) occupy a significant fraction of public attention to CPUs, yet embedded and mobile devices are a large fraction of the world processor market. A significant fraction of that market is highly energy-sensitive.

Accordingly, processors suited to a wireless platform are likely to draw on experience with implementing power management in designs like the ARM and PowerPC ranges (especially those targeted at mobile markets).

While the Intel 32-bit architecture is in many respects a poor design for mobile computing, even that processor family has had significant work put into power management, to support notebooks. However, a design more

²<http://www.eembc.org/>

specific to energy efficiency is a better starting point, especially as the wireless market has the opportunity to start afresh.

RISC designs in general are easier to target to energy efficiency, because it is easier to split their pipelines into non-interacting stages. An irregular instruction set, as soon as more than one instruction per cycle needs to be decoded on a cycle, results in a tight dependency between the fetch and decode stages, because it is not possible to know that exactly n instructions (for n -way fetch and decode) have been fetched until some decoding is done. Such dependences between stages of the pipeline make it hard to minimize the amount of parallel logic, which drives up energy requirements.

Of the mainstream (as in desktop and server markets) RISC designs, the PowerPC has possibly achieved the highest penetration in mobile and embedded markets, because of its adoption by Motorola. Of processors without a significant desktop or server market, ARM likely has the significant penetration by virtue of having been early to specialize in energy-efficiency.

Processors addressing the mobile market generally come in a wide variety of variations to suit cost, energy and packaging concerns. A case has even been made for application-specific processor design to reduce power needs [3].

2.1.3 Challenges in Retargeting Conventional Designs

In 1988, some DSP advocates were predicting that DSPs would soon put supercomputers out of business [20]. What actually happened was that microprocessors based on RISC principles (and to a lesser extent mass-market designs) put severe pressure on supercomputers. We therefore cannot discount the potential of a good general-purpose design in a niche market.

However, “niche” needs to be qualified. Supercomputers, while a small market, require considerable investment in compiler technologies, optimization of specific applications and general expertise in performance tuning across a diverse if small community of users, many of whom are programmers. Embedded devices or special-purpose devices are a different kind of niche. Like supercomputers, the base of developers is relatively small and specialized, but the non-programming users can be a relatively large number of people. The applications are also small and often simple enough to justify retargeting for a speed or cost gain. A closer analogy to the wireless platform is high-end graphics controllers in the era before they could be put on a single chip. In the 1990s, Silicon Graphics (SGI) went through a series of revisions of their high-end graphics boards which covered a wide range of supercomputer processor architectures of the day. They started with a pipelined architecture, went to a SIMD (single instruction multiple data stream) design [21], and ended up with a shared-memory design – sometimes supporting multiple completely different designs for different price points [22].

SGI was able to vary the underpinnings because the software architecture (GL, later OpenGL) was kept consistent.

When we look at the wireless platform, the design trade-off of choosing what to fix and what to leave open applies. In the SGI case, the programming interface was kept consistent (at binary level): a hardware abstraction layer made all these variations in architecture transparent (except for performance and some advanced functionality which was performance or memory dependent).

To put all this together, the nature of the niche we are dealing with suggests a hardware abstraction layer to hide specialized processing which would be useful to reimplement as needed to address new requirements. A specialized processor (DSP for example) artificially creates a boundary where functionality on that specialized processor is seen as separate, and not substitutable. If that boundary could be hidden, as with the SGI example, changing the underlying implementation could be seen as a performance optimization, rather than a major redesign.

A challenge with using a conventional design in the wireless platform is therefore to maintain this distinction between the mutable and immutable aspects of the architecture without the (possibly artificial) dividing line of what runs on the conventional processor versus the special-purpose design. Rather, logical divisions need to be established, in which it is clear which components are likely to change over time, and these components should be isolated by an abstraction layer.

Once this high-level architectural issue is disposed of, it becomes possible to focus on performance issues.

Given cost constraints, using a single processor has significant value. Even if performance goals are harder to meet, there is some value in attempting to implement DSP functionality on a conventional design. The big obstacle is the fact that hard real time requirements are central to many applications. Being within 10% of the performance of a more expensive design may generally be a useful goal in cost-sensitive markets, but if a real time deadline is missed as a consequence, the design is no good.

The two major issues identified here for conventional designs, therefore, are defining an architecture which clearly provides an abstraction layer to hide aspects where implementation should be flexible, and to ensure that real time design constraints can still be met with a less specialized design.

2.1.4 Hybrids

Given that a completely conventional design could run into performance and design problems, while reducing the component count is a useful goal, it is worth considering hybrid designs.

Such hybrids exist in various forms, including vector extensions to existing instructions sets, as in Motorola's AltiVec extensions [23] to PowerPC processors. There are also processors with more than one core, such as designs from Texas Instruments which include an ARM processor core and one of their own DSP cores [24].

As experience with AltiVec has shown, a hybrid pipeline is difficult to scale up in speed. Further, integrating DSP-oriented extensions into an instruction set blurs the boundary between the application binary interface (ABI) and the low-level implementation. In the model we are working towards, it would be useful to maintain a distinction between operations central to computation and operations central to communication and I/O. This distinction is easier to make if the low-level processing of these functions is not too tightly integrated into the CPU – which suggests a multiple core design.

Another advantage of a multiple-core approach is that it makes power management simpler. If a core is not being used, provided the latency of restarting it is acceptable (e.g., for meeting real-time deadlines), it can be turned off when it is not in use. While parts of any CPU can in principle be turned off when not in use, a more integrated design reduces the opportunities for limiting energy use. For example, parts which relate directly to interfacing with the pipeline may not be possible to turn off, and it would not be feasible to turn off the parts of a cache used by an idle functional unit (or other component).

Finally, different optimizations may be easier with a decoupled design: if the conventional core is running more or less constantly whereas the DSP core is running sporadically with bursty behaviour, different peak energy use may be tolerable in each core – possibly leading to approaches like running at different voltages, which would be much harder to achieve in a monolithic design.

2.2 Summary

Processor design for mobile computing has a number of challenges. If the right trade-off between precise specification and flexibility can be found, a design covering a wide range of applications and with a relatively long lifetime should be possible.

Low energy should remain a priority. The role of specialized instruction sets as in DSP designs requires more detailed examination. The question as to whether DSP instruction sets represent a significant improvement over an aggressive conventional design is more complex to investigate than “conventional” architecture studies, because of the need to take into account hard realtime requirements, which make a traditional Amdahl's Law-style investigation (measure the average over a whole run to determine speedup) problematic.

2.3 Operating Systems

Given that one of the challenges in wireless computing is extending the platform to new niches, cost is likely to continue to drive memory requirements to much smaller sizes than are acceptable in desktop and server computers. For this reason, operating systems will continue to be required to have small memory footprints. Energy efficiency will also have to be a more significant aspect of operating system design [5].

Historically, designing for small memories has led to designs which have been abandoned as newer, larger memories became affordable (for example, the move from the PDP-11 to the VAX). A change becomes difficult when a new memory system goes beyond limits imposed by the instruction set architecture or the software.

Given the scalability model in 1.2, it is useful to think in terms of an operating system which works with very small configurations, but which can also scale up. A microkernel with very fast messaging to support building extra layers outside the kernel is one potential model.

Such operating systems exist, for example, L4Ka [25], which is a small kernel, with emphasis on supporting traditional kernel functionality outside the kernel efficiently. The flexibility of the design, speed of messaging and small size of a minimal kernel make L4Ka and derived kernels candidates for a wireless platform. Not only do they support very small configurations, but the design is intended to scale to very large ones. The key is modularity:

components which are not needed can be left out. Given the relatively high-speed interface to the kernel, most components are not part of the kernel, which makes it relatively easy to add or subtract functionality.

2.4 Network Protocols

Network protocols for wireless systems are possibly the area of greatest challenge, because interoperability is most strongly dependent on ability to communicate. The biggest challenge is to design a flexible fabric on which protocols can be built, so backward and forward compatibility become possible.

Because one of the strengths of the wireless device area is relatively rapid evolution of new protocols, it is important to try to find models for allowing rapid change, without obsoleting devices whenever possible.

One strategy would be to implement as much as possible in software. Software-defined radio (SDR) is a broad catch-all term for moving as much of the traditional hardware layer of wireless communication into software as possible. Potential problems include regulatory complications and security [26]. In general, wireless protocols can introduce security problems of their own; while fixing these problems may drive defence and attacks to other layers [27], the diversity of wireless protocols requires continuing reevaluation of security issues.

The general notion of mutable protocols has a lot of appeal – as long as the physical device was able to operate at a given frequency, and there was some low-common-denominator basis for setting up communication, a device lacking a given protocol could in principle ask the wireless network for the missing software. This kind of functionality has more appeal in the wireless than the wired world, as the mobility of devices is likely to lead sooner or later to encountering a situation where protocols which may at first have appeared optimal are no longer the best choice (possibly even not working with the new environment).

Resource discovery of this kind would require close attention to security issues. A kernel with fine-grained protection, capable of minimizing possibilities for malicious software (even low-level drivers) would be desirable.

2.5 Summary

Processor architectures specific to requirements of wireless computing, particularly DSPs and low-energy designs, are crucial to developing a standard platform. The platform needs to allow flexibility for change. One approach to supporting this flexibility is to implement multiple-core processors, with communications and processing separated. In this way, the flexibility needed to move rapidly to new protocols can be divided off from the processing requirements of the application.

An operating system to support the required flexibility should be highly modular in design, with capabilities of adding or subtracting functionality. It should also have sufficiently fine-grained protection to allow potentially untrustworthy drivers to be used, as part of resource discovery.

3 Problems and Challenges

Designing a wireless platform requires careful balance of fixed versus mutable aspects of its design. The lack of standards (or fluidity of standards) in the wireless world is both a strength and a liability. Building on the strengths while limiting the weaknesses is a challenge for system architects. A poor initial choice could constrain future designs in the wrong areas, and reduce the value of a standard platform.

In the processor area, there is a temptation to limit address space because of currently small memories being used in many mobile devices. However, demands of pervasive and ubiquitous computing applications could lead to large-scale shared address spaces with persistent objects. Given the open-ended nature of the application space, choosing suitable processor design features is more difficult than in traditional niches, where the scope of applications has generally been clearer. For the same reason, instruction set design trade-offs are not clear: a tightly-encoded instruction set designed to minimize memory use at the expense of speed may seem a good idea if all devices are small, but speed may be of the essence in some applications (for example, speech recognition).

The ideal scenario would be to have flexibility in instruction set design. A possible implementation would be to crack instructions into simpler operations which were implemented directly in hardware, with a simplified front end to translate the instructions to the internal format – an approach already in use in some commercial designs [28]. In cases where speed was the major requirement, and instruction set architecture close to the machine instructions could be used. Where memory use or communication bandwidth was more of a concern, a more tightly encoded instruction set could be used, with more steps to crack instructions into machine operations.

The biggest difficulty with operating systems is the lack of accepted standards. While a tightly-coded microkernel-based system would be ideal, existing operating systems already have market share, which could prove a stronger issue than technical suitability. However, given that the application base is small (a small number per device that is), moving to a new platform may not be so difficult.

The networks area, while presenting some of the most difficult problems, builds on the other areas. If a good processor architecture can be found with strong support for communications software, greater flexibility in network protocols could result. Similarly, if an operating system with strong support for modularity and fine-grained protection is used, network protocols can be flexible, with options like resource discovery to add new protocols on the fly.

4 Conclusions

Defining a wireless platform presents many challenges, but if the promises of ubiquitous and pervasive computing are to become reality, something better than a range of ad hoc, incompatible device designs is needed. As has been noted in the automotive industry [11], a common platform is the way to go.

This section wraps up the discussion by summarizing the key issues raised before, proposing a way ahead, and providing an overall conclusion.

4.1 Summary

Wireless devices (most but not all of which are mobile) are a growing field. The flexibility inherent in being able to operate without wires is a great strength of the area, yet an attribute which has the potential to prevent integration of services and applications.

A standard platform has the potential to provide a clean divide between the aspects of wireless devices which are useful to standardize, and those which are better left open.

Processor design presents a number of challenges, arising out of the fact that what is done in software and what is not can change over time, and the fact that the scale of computation is likely to change over time. Approaches like multiple cores and flexible instruction sets seem worth exploring. Maintaining an architectural split between mutable and fixed aspects of the architecture seems worthwhile. Whether this is a hardware or a software split is to some extent a matter of design philosophy, but a hardware split has the advantage of allowing aggressive pursuit of design alternatives in areas where frequent change is expected.

Operating systems which scale from very small to very large systems are a useful addition to the platform, as are highly flexible approaches to designing network protocols, with less emphasis on a fixed division between hardware and software layers. Design for energy efficiency should also become a significant operating systems issue.

Networks are one of the areas of greatest challenge, because wireless operation is inherently more flexible than wired operation. Blurring the hardware-software boundary is a likely consequence of improved low-cost processors.

4.2 Way Ahead

To take these ideas forward, it would be useful to provide a baseline for future evaluation. A set of benchmarks for desired behaviours, especially taking into account real-time requirements, would be useful. Application-based benchmarks are likely to be hard to pin down because of the wide range of devices. Scalable benchmarks for real-time behaviours would be useful (perhaps drawing from the idea of scalable transaction processing benchmarks [29]).

Once a basis for comparison is established, it would be useful to define a relatively abstract sample device, which could easily be modified to represent a range of different application styles. This sample device could become the basis for future simulators and implementations.

Given an abstract sample device, it becomes possible to define a range of alternative implementation strategies, which can be measured against more specific needs.

4.3 Overall Conclusion

The ideas described here are a starting point: much detail still needs to be worked through. The notion of a standard platform for wireless devices has attractions, and, if it is to be done, should be done as well as possible, to provide a significantly better base than no platform at all. Given that some flexibility is inherently given up in going for a platform rather than an ad hoc design, it is important to be sure that the end result confers more advantages than the costs.

If a good division between standardization and flexibility can be found, a design which will be good for several decades is possible, if history of other successful architectures is a basis on which to judge.

Acknowledgements

This paper was inspired by a visit to Trevor Mudge at University of Michigan. The trip was funded by a grant from the University of Queensland.

References

- [1] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto. An instruction-level functionally-based energy estimation model for 32-bits microprocessors. In *Proc. 37th Conf. on Design automation*, pages 346–351, Los Angeles, CA, 2000. ACM Press.
- [2] Flavius Gruian and Krzysztof Kuchcinski. LEnes: task scheduling for low-energy systems using variable supply voltage processors. In *Proceedings of the conference on Asia South Pacific Design Automation Conference*, pages 449–455, Yokohama, Japan, 2001. ACM Press.
- [3] Chris Weaver, Rajeev Krishna, Lisa Wu, and Todd Austin. Application specific architectures: A recipe for fast, flexible and power efficient designs. In *Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (CASES01)*, pages 112–115, November 2001.
- [4] Nam Sung Kim, Krisztin Flautner, David Blaauw, and Trevor Mudge. Drowsy instruction caches: leakage power reduction using dynamic voltage scaling and cache sub-bank prediction. In *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, pages 219–230. IEEE Computer Society Press, 2002.
- [5] Tao Li and Lizy Kurian John. Run-time modeling and estimation of operating system power consumption. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 160–171. ACM Press, 2003.
- [6] R. Kumar, K. Farkas, N.P. Jouppi, P. Ranganathan, and D.M. Tullsen. Processor power reduction via single-ISA heterogeneous multi-core architectures. *Computer Architecture Letters*, 2(1):2–5, July 2003.
- [7] Liang Cheng and Ivan Marsic. Piecewise network awareness service for wireless/mobile pervasive computing. *Mobile Networks and Applications*, 7(4):269–278, 2002.
- [8] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):29–58, 2000.
- [9] Steven M Cherry. Broadband a go-go. *IEEE Spectrum*, 40(6):20–25, June 2003.
- [10] Robert Tolksdorf and Dirk Glaubitz. XMLSpaces for coordination in web-based systems. In *Proc. Tenth IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 322–327, Cambridge, MA, June 2001.
- [11] Alberto Sangiovanni-Vincentelli. Electronic-system design in the automobile industry. *IEEE Micro*, 23(3):8–18, May/June 2003.
- [12] Guruduth Banavar, James Beck, Eugene Gluzberg, Jonathan Munson, Jeremy Sussman, and Deborra Zukowski. Challenges: an application model for pervasive computing. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 266–274, Boston, Massachusetts, United States, 2000. ACM Press.

- [13] David Gifford and Alfred Spector. Case study: IBM's system/360-370 architecture. *Communications of the ACM*, 30(4):291–307, 1987.
- [14] Steven J Vaughn-Nichols. OSs battle in the smart-phone market. *Computer*, 36(6):10–12, June 2003.
- [15] David A. Patterson. Reduced instruction set computers. *Communications of the ACM*, 28(1):8–21, 1985.
- [16] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kauffmann, San Francisco, CA, 3rd edition, 2002.
- [17] M. Prasad, P. D'Arcy, A. Gupta, M. Diamondstein, and H. Srinivas. Half-rate GSM vocoder implementation on a dual Mac digital signal processor. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP '97)*, volume 1, pages 619–622, April 1997.
- [18] TI. *TMS320C6000 CPU and Instruction Set Reference Guide*. SPRU189F. Texas Instruments, October 2000.
- [19] Neil Bergmann, Peter Waldeck, and John Williams. A catalog of hardware acceleration techniques for real-time reconfigurable system on chip. In *Proc. 3rd IEEE Int. Workshop on System-on-Chip for Real-Time Applications (IWSOC'03)*, pages 112–115, Calgary, Alberta, Canada, 30 June–2 July 2003.
- [20] L. Robert Morris and Stephen A. Dyer. Floating-point digital signal processing chips: The end of the super-computer era? *IEEE Micro*, 8(6):86, November 1988.
- [21] Chandlee B. Harrell and Farhad Fouladi. Graphics rendering architecture for a high performance desktop workstation. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 93–100. ACM Press, 1993.
- [22] Mark J. Kilgard. Realizing OpenGL: two implementations of one architecture. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 45–55, Los Angeles, California, United States, 1997. ACM Press.
- [23] Motorola. *AltiVec Technology Programming Environments Manual*. Motorola, 2001.
- [24] TI. *OMAP5910 Dual-Core Processor*. SPRS197A. Texas Instruments, August 2002.
- [25] Uwe Dannowski, Kevin Elphinstone, Jochen Liedtke, Gerd Liefländer, Espen Skoglund, Volkmar Uhlig, Christian, Ceelen Andreas, and Haerberlen Marcus Völp. The L4Ka vision. Technical report, University of Karlsruhe, System Architecture Group, April 2001.
- [26] SDR Forum. Report to FCC on issues and activities in the area of security in SDR. Technical report, SDR Forum, September 2002.
- [27] Nancy Cam-Winget, Russ Housley, David Wagner, and Jesse Walker. Security flaws in 802.11 data link protocols. *Com. ACM*, 46(5):35–39, May 2003.
- [28] J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy. POWER4 system microarchitecture. *IBM Journal of Research and Development*, 46(1):5–25, 2002.
<http://researchweb.watson.ibm.com/journal/rd/461/tendler.html>.
- [29] TPC. *OLTP Benchmark Proposal Development Guidelines*. Transaction Processing Performance Council, 4 December 2001.