Trends in Hardware Architecture for Mobile Devices

Dr. Margarita Esponda
Institut für Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin

November 2004

Abstract

In the last ten years, two main factors have fueled the steady growth in sales of mobile computing and communication devices: a) the reduction of the footprint of the devices themselves, such as cellular handsets and small computers; and b) the success in developing low-power hardware which allows the devices to operate autonomously for hours or even days. In this review, I show that the first generation of mobile devices was DSP centric – that is, its architecture was based in fast processing of digitized signals using low-power, yet numerically powerful DSPs. However, the next generation of mobile devices will be built around DSPs and low power microprocessor cores for general processing applications. Mobile devices will become data-centric. The main challenge for designers of such hybrid architectures is to increase the computational performance of the computing unit, while keeping power constant, or even reducing it. This report shows that low-power mobile hardware architectures design goes hand in hand with advances in compiling techniques. We look at the synergy between hardware and software, and show that a good balance between both can lead to innovative low-power processor architectures.

1 Mobile and Embedded Computing

1.1 The Mobile Revolution

The number and variety of mobile devices has increased steadily in the last ten years. Cellular telephones, first introduced in the 1980s, have achieved a penetration rate of almost 70% in some countries and are certainly the main factor behind the current explosion in the use of mobile devices. However, there are other types of mobile devices, such as:

- Personal Digital Assistants (PDAs)
- MP3 Personal Players
- Point-of-Sale mobile devices (for credit card payments)
- Mobile CD Players
- Portable GPS Navigation systems
- Digital cameras

- Digital video cameras
- Subnotebook computers
- Pen digitizers (such as the Logitech io Digital Pen)
- Handheld OCR systems

Although there is by now a real zoo of mobile devices, with species and subspecies of them, there are three main features common to most them:

- Mobile devices provide wireless communication to a base station (a personal computer, or a whole telephone, or data network);
- Mobile devices must be small to be portable;
- Mobile devices must remain operational for several hours or days without a battery recharge.

We can say that communication, performance, and low-power operation are the main factors affecting the design and deployment of mobile devices. The mobile revolution is just beginning and it is necessary to consider all the implications that it will have for hardware architects [Intel 2003], [Koushanfar 2000]. Until now, academic research has concentrated solely on performance, because the driving motor behind the growth of the computer industry have been sales of desktop systems (Pentium or PowerPC based computers). Most hardware architecture textbooks usually discuss only high-performance systems [Patterson 2002], [Tanenbaum 1998]. However, embedded and mobile systems have started to eclipse traditional computers in terms of the number of units sold, and this makes the more important to look at new hardware architectures for mobile devices.

1.2 Ten Year Cycles in Computer Design

The primary motor behind computer architecture evolution has been, until now, Moore's Law. The circuit density of digital chips doubles every 18 months. Thus, the number of digital functions that can be put on a chip has increased exponentially since the invention of the integrated circuits, while their price has fallen at the same exponential rate. The computer industry has gone through several generations of computing machines, but. curiously, the expansion and dominance of every kind of technology has lasted about ten years. We have had

- The 1960s as the period of dominance of large computers (*mainframes*),
- The 1970s, with the emergence and dominance of *microcomputers*,
- The 1980s, when the *personal computer* was brought to the masses of consumers,
- The 1990s, when the stand-alone computer was *networked* and the global information highway emerged,
- And the 2000s, the current decade, in which microprocessors for *embedded* or *mobile applications* constitute the bulk of the computing power being sold [Wolf 2004].

If we can extract a lesson from the past of computer technology, then it would be this: the dominant technology in every decade has been introduced earlier (microprocessors, for example, had been available since the 1970s), but its dominance has always been based in one initial "killer application" which has stimulated demand, higher sales, and increased production of the base technology, which in turn has led to reduced costs. In the electronics industry, a mass market and low prices combine together into a positive feedback loop, which

in turn leads to improved devices and increased performance. At some point, miniaturization opens the way for even smaller devices, and a new round of innovation starts.

The “killer application” of the 1990s was networking, connectivity to the Internet. Until then, the personal computer was a substitute for the typewriter, or a machine for the office, or a computerized accountant. With the Internet, the personal computer became a new and pervasive communication medium, a global library, and a window to the world. It became a household item which cannot be missing in modern homes, at least in the industrialized world.

Economists have developed the concept of “network economies” in order to analyze such objects whose usefulness actually increases with the number of units sold. A toaster, for example, does not exhibit any kind of network economy. If my neighbor has the same toaster or not, it is irrelevant for me. But a computer is different: my computer becomes more valuable if it is connected to all my neighbors. If we have n persons, and they all can interconnect, we then have a total of n^2 connection paths. As Vincent Cerf has put it: the usefulness of a computer increases quadratically with the number of persons connected to the network.

The telephone network is the classical example of a system with networking economies in action. Therefore, it comes as no surprise, that the cellular wireless telephone has been, until now, the driving force behind the mobile revolution. The number of subscribers to the cellular systems has increased exponentially in the last few years, and now we look at around 1500 million cellular subscribers in the world. Although there are large regional variations and Africa remains being a special case, almost a fourth of the world population is now connected to the cellular network. The world has shrink one step further due to the mobile revolution.

1.3 Embedded Processors and Mobile Communication Statistics

Some numbers can help to bring some perspective into our discussion.

First, the number of microprocessors sold in the world is now dominated by communication and embedded applications. Embedded microprocessors are now present in TV sets, DVD recorders, hi-fi sets, microwave ovens, washing machines, refrigerators, and in cars. A top of the line BMW features some 60 to 70 microprocessors which are used to control the brakes, the air bags, the windows, etc. It has been calculated that every American household owns around 60 microprocessors embedded in mobile devices and in household items.

Desktop microprocessors are built for raw computing performance and computers become now obsolete after only three years. The fastest chips dominate the market and the slower chips disappear. But in the case of embedded microprocessors there is a much wider margin of applications. The fastest microcontroller in the 4 to 32 bit range can be 1000 times faster than the slowest. There is much more variety in the microprocessor market than in the high-end market, because there are many more different kind of applications.

An estimate made by Hennessy in 1999 of the number of embedded and PC processors sold until 1998, as well as a projection for the later years, came to the conclusion that PC processors would constitute only 25% of the market by 2004 [Hennessy 1999]. The reality, however, has become much more pointed. According to the World Semiconductor Trade

Statistics, there are more than 5000 million embedded microprocessors in use today, and in terms of units sales Intel, AMD, IBM and Motorola, account for only 6% of the world market. That means that 94% of the microprocessors sold today go into embedded or mobile applications. Fig. 1 shows a graph of the monthly unit sales of 4, 8, 16, and 32 bit microprocessors. As can be seen, five times more 8-bit microprocessors are sold than 32-bit designs. All these small microprocessors go into embedded applications.

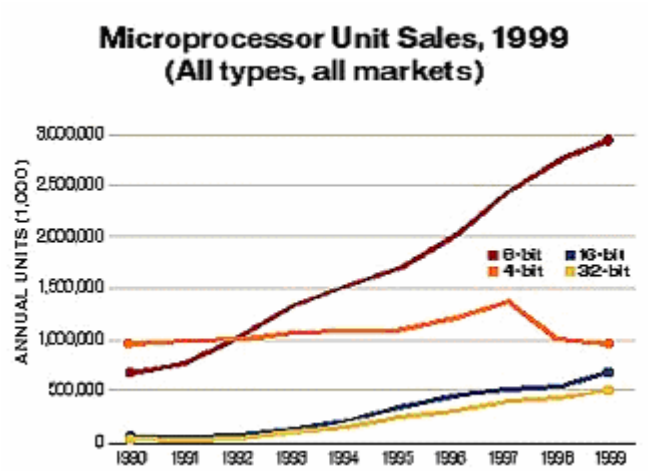

Figure 1: Embedded processors dominate the sales of microprocessors [Turley 2002]

Second, as mentioned before, the current mobile killer application is telephony. Cellular telephones have the primary function of allowing people to get in contact, although many of them also provide data services, such as MSM messaging, e-mail, and web browsing. Fig. 2 shows the number of digital cellular subscribers in the world. In the summer of 2004, there were around 1500 million digital subscribers, and 1108 million of them were using GSM devices. Other technologies account for the rest of the subscribers.

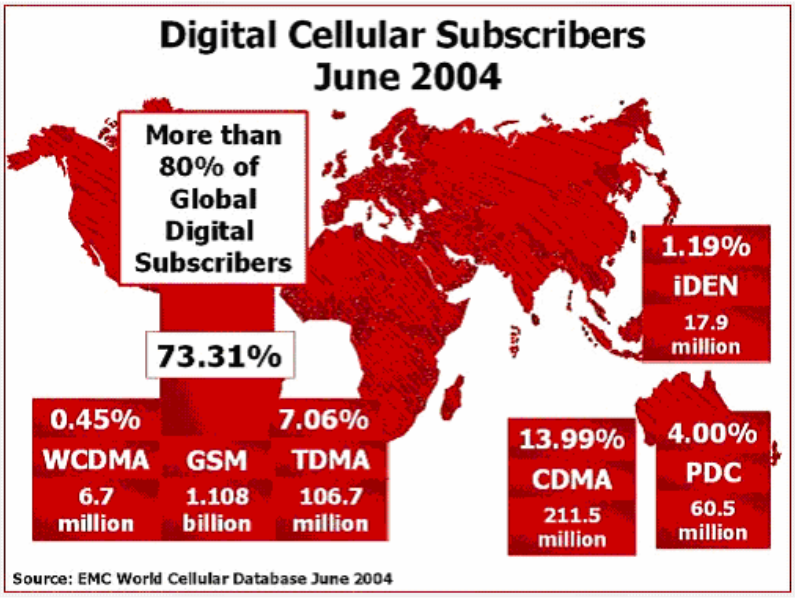

Figure 2: World Digital Subscribers (image from www.3gamericas.org)

As can be seen in Fig. 3, a fourth of the digital subscribers are located in Europe, but almost 40% in Asia. The USA has lost the lead in cellular telephony, although analog cellular telephones were first deployed there.

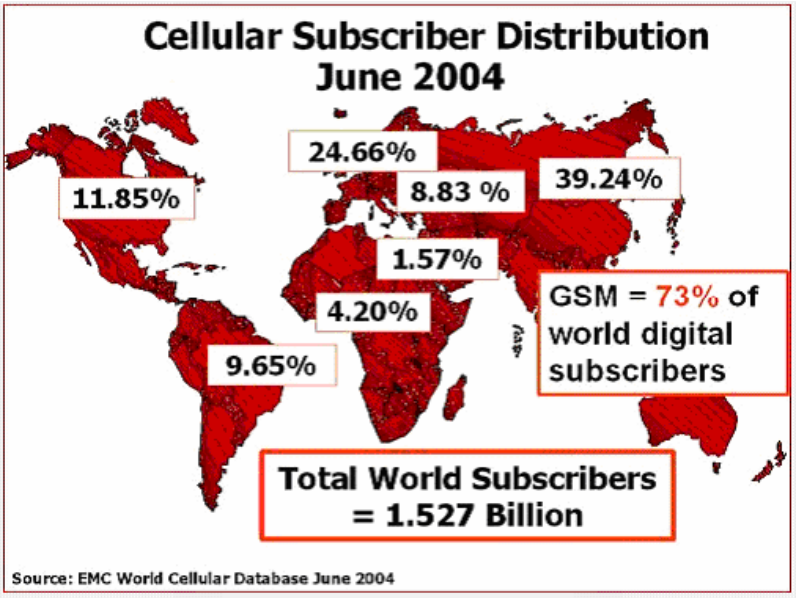

Figure 3: Distribution of Digital Subscribers (image from www.3gamericas.org)

An interesting example of the importance of cellular telephones is Fig. 4. Digital cameras are substituting the traditional film-based devices, but most of the future planned growth is for digital cameras installed in cellular telephones. “Smart telephones”, which integrate several functions, such as MP3 player, digital camera, and PDA will slowly substitute the traditional voice-only systems. The cellular telephone will become a data central and the hardware architecture will change accordingly.

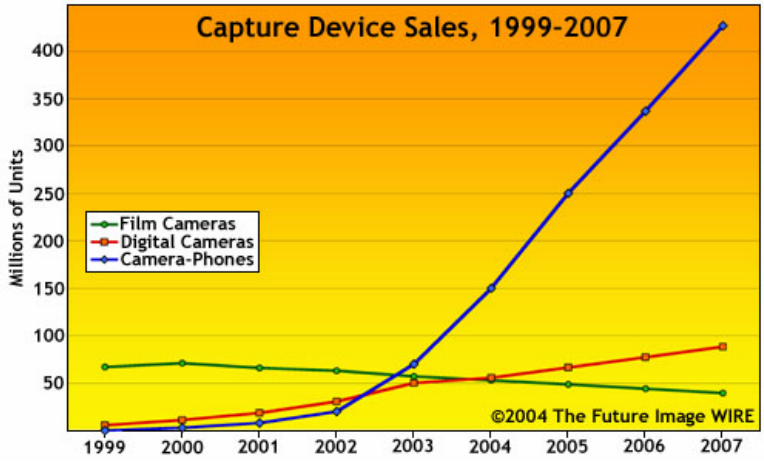

Figure 4: Most of projected sales of digital cameras will be embedded in cellular telephones.

2 DSP centric architectures and 2G systems

It has been said, that Digital Signal Processors (DSP) are to cellular telephones what the microprocessor is to desktop systems, that is, the heart of the whole design. In this section we examine this assertion and how DSP architectures have evolved in the last years.

DSPs became important with the transition from analog to digital cellular telephony. The first generation of cellular systems (called 1G systems) used analog transmission and was started commercially mainly in the 1980s. Analog systems had the disadvantage of requiring more

power for transmission, they allowed less users in the same band interval, and the handsets were usually bulky.

In the early 1990s, analog cellular telephony was gradually substituted by second generation (2G) systems. The signal was not transmitted in analog form, it was previously converted into sequences of bits encoding the signal. Digital transmission allows the handset to compress the voice data, saving bandwidth. More complex control and connection protocols can be handled, and this allows to accommodate more users in the same frequency band. Moore's law, on the other side, allowed companies to build handsets which were smaller and smaller. The real explosion of cellular telephony started only with this second generation of digital devices.

The motor behind digital telephony are the DSPs. The first DSP was introduced by ATT&T in 1979, but the most successful design came from a chip company, Texas Instruments. Ever since then, DSPs from Texas Instruments have dominated the market for digital telephony, and today 65% of the digital telephones use a chip made by TI. In what follows, we review why DSPs are so special for streaming applications. We show that 2G telephony systems are DSP centric, that is, their architecture is organized around a DSP chip with some memory and peripherals. Fig. 5 shows that the market for DSPs is driven by wireless applications. Although DSPs have found other niches, digital telephony is still their main niche.

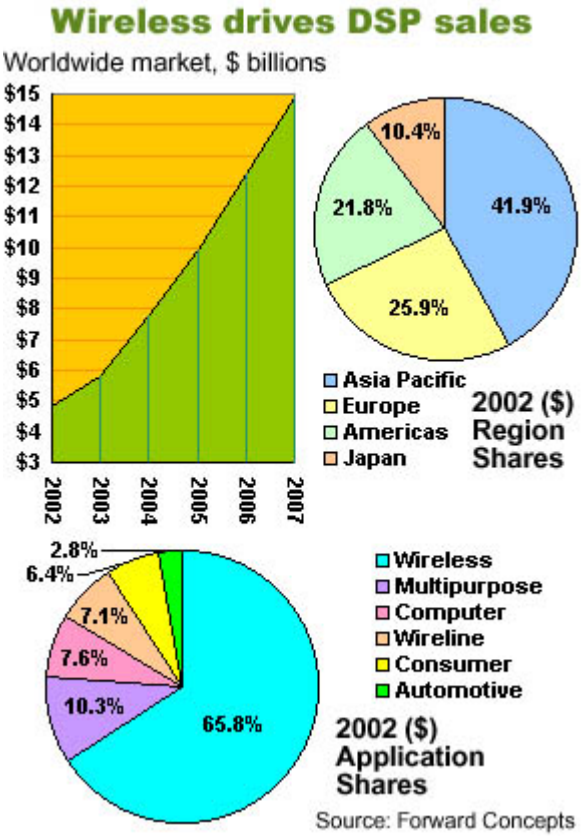

Figure 5: The DSP market is driven by wireless applications (image from www.semireporter.com)

2.1 Digital Signal Processors

Digital Signal Processors are microprocessors built with one main application in mind: the fast processing of digitized signals for maximizing throughput. A digital signal can be analog

microphone data which has been transformed into a stream of bytes for processing and transmission. A digital signal processor is very much the equivalent of an analog circuit in which a signal comes in, and leaves the circuit transformed, after a small delay. Analog circuits operate in real time – DSPs must do the same. However, signal processors are much more versatile than analog circuits. DSPs can be programmed and variations of an algorithm or filter can be implemented with the same chip. DSPs can compute Fourier transforms, cosine transforms, smooth signals or compress them. Only the software has to be adapted, unlike analog circuits where the whole circuits has to be redesigned.

DSPs can be thought as universal replacements for analog signal processing circuits. Their design has not been optimized for classical data processing, such as data bank search or spread-sheet computation. DSPs excel when the same operation has to be applied repetitively to a large incoming stream of data which leaves the processor almost immediately. An MP3 player, for example, is a good example. The DSP receives the MP3 stream, decodes it and generates the wave signal for the headset. There is almost no logic involved: the MP3 stream is converted continuously in an output stream.

Since DSPs have been optimized for certain applications, their architecture can also be specialized. Traditional hardware architecture rules do no apply in their case. Modern desktop processors, for example, almost universally use a load-store architecture. This means that a multiplication, for example, involves the following steps:

- copy address of first operand into a register
- load first operand
- copy address of second operand into a register
- load second operand
- multiply

A DSP can perform all these operations in a single cycle. It can do even more: it can update the argument registers to point to the next arguments in memory and can accumulate the result of the multiplication to a previous temporal result. The scalar product of two vectors can be computed with a DSP in such a way that two vectors of length n are multiplied in n cycles.

DSP processors therefore, exhibit some characteristics of RISC processors, but not all of them. DSPs are RISCs because they use deep pipelines, but they have many CISC type instructions which are efficient since they are very frequently used.

This the main design challenge with specialized processors, such as DSP, that is, finding the optimal set of special instructions which maximizes their speed and throughput. In the next section we loom at traditional DSP design and which operations have been traditionally sped up.

2.2 Traditional DSP Design

In this section we discuss some of the design characteristics which put DSPs apart from other kind of microprocessors.

2.2.1 Harvard Architecture

DSPs are designed for throughput. Digitized signals must go through the processor as fast as possible. Since at least an instruction is executed in every cycle, it is important to have a processor-memory bus for the program and one for the data. This is called a Harvard architecture, in remembrance of the Mark I computer built in 1944 at that university. The program was stored in punched tape and the data in memory units, that is, there were two separate communication channels between processor and memory.

A Harvard architecture requires two output buses for the addresses generated for the processor: a data address and a code address. It requires additionally the read bus for code and a read/write bus for the data. The address bus for the code can be of limited width (24 bits, for example) since it is assumed that the code to be executed is not as large as in desktops. The read/write data bus can be designed to read a 32-bit word, but each 16-bit half-word can be treated internally as a separate argument. This allows the DSP to load two arguments into the processor in a single cycle. Fig. 6 shows a diagram of a canonical Harvard architecture.

In old simple DSP designs there is no cache for the instructions loaded into the processor, but there is a variety of buffers which can be used to speed up computations.

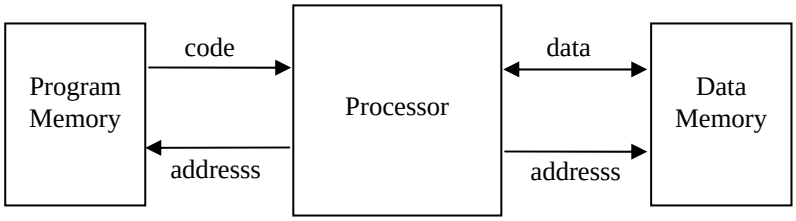

Figure 6: The canonical Harvard Architecture

In order to speed up access to memory, DSPs use registers which can be set and modified by address generation units. Digital filters, for example, are implemented by multiplying a vector of *n* coefficients with *n* digitized data points. The coefficients remain constant while the data is processed one window of length *n* at a time. It is then useful to store the *n* coefficients in a circular buffer, in which the pointer to the data returns to the beginning of the buffer when all coefficients have been used. It would be possible to perform this operation using indirect addressing and general-purpose registers, but since the operation is so common, a better approach is to have special registers where the necessary addresses are generated automatically after each data load. Updating the address registers can be overlapped with the rest of the operations and their update does not consume time.

DSPs can have several different kinds of address units, for code or data. The motivation is always to speed up convolutions, that is, digital filtering.

2.2.2 Multiply-Accumulate ALU

The kind of instruction most commonly associated with DSPs is *multiply-accumulate* (MAC). When computing scalar product of vectors of length *n*, *n* pairs of numbers have to be multiplied pairwise and the partial results are added in a final result. In general purpose

processors this is done using n multiplications and n additions, with load and store instructions in between. In a DSP the data path is laid out in such a way that the output of the multiplication unit connects directly to an adder, which adds and saves all partial results. In this way an scalar product of vectors of length *n* takes only *n* cycles. The MAC unit works in conjunction with the address generation units, so that in each cycle the two arguments needed are loaded directly from memory. Fig. 7 shows a diagram of a classical MAC unit.

Most DSP use fixed-point arguments, although some have floating-point units. The reason for the use of fixed-point, is that in image and voice processing the data seldom has a resolution higher than 16 bits, and in most cases limited numerical precision is sufficient. The data does not vary across 128 orders of magnitude, as it can do in floating-point format.

The use of fixed point arithmetic is also important because it is straightforward to build one-cycle multiplication units for 16-bit fixed-point data. One of the arguments is shifted 16 times using special hardware and the 16 shifted numbers, plus the last partial accumulated result, are XORed in one cycle. The carrys are also computed, and are passed to the addition unit, which computes the final result using carry-lookahead techniques.

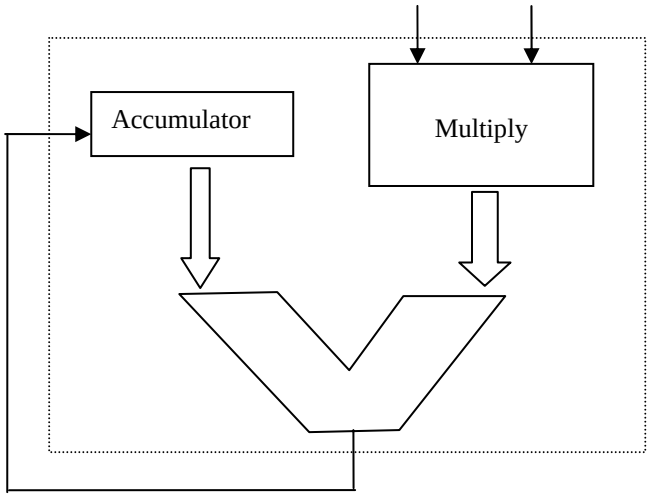

Figure 7: A Mac unit can multiply two arguments and accumulate the result in one cycle

2.2.3 Zero-Overhead Looping

In digital signal processing there are operations over streams which can be implemented as loops of instructions. The loop is of fixed length and ie executed completely. Instead of testing a condition and branching if the end of the loop has not been reached, DSPs provide sets of special registers which count the iterations, are incremented automatically after each one, and which move the PC to the beginning of the loop when the loop body has been processed. The DSP execution units sees a linear flow of code without branches. This effect, which is achieved in RISC processor with the help of branch prediction and speculation, is achieved at a lower cost using the looping instructions of DSPs.

DSPs provide many other special instructions, useful only in signal processing, such as those needed for Viterbi filtering, etc.

2.3.4 Prototypical Architecture of a Mobile System

Fig. 8 shows a diagram of a typical cellular system of the second generation. It is useful for us because from this diagram we can deduce which kind of instructions and capabilities are needed in a cellular telephone.

As the diagram shows, there is a radio module which handles reception and transmission of digital signals. The RF interface handles the conversion of analog radio waves into bits, or vice versa for transmission. The bit stream is then passed to the DSP for speech or data decoding. The DSP can process the stream all by itself or can receive help from specialized chips (ASICs). A small microcontroller handles the user interface, the keyboard and LCD display, and the small operating system which orchestrates all services in the cellular handset. A flash memory is part of the system. It can be used to store numbers, addresses, and other information. The audio interface handles the microphone and speakers. The kernel of the system is the DSP, as can be seen. A battery provides power to the system.

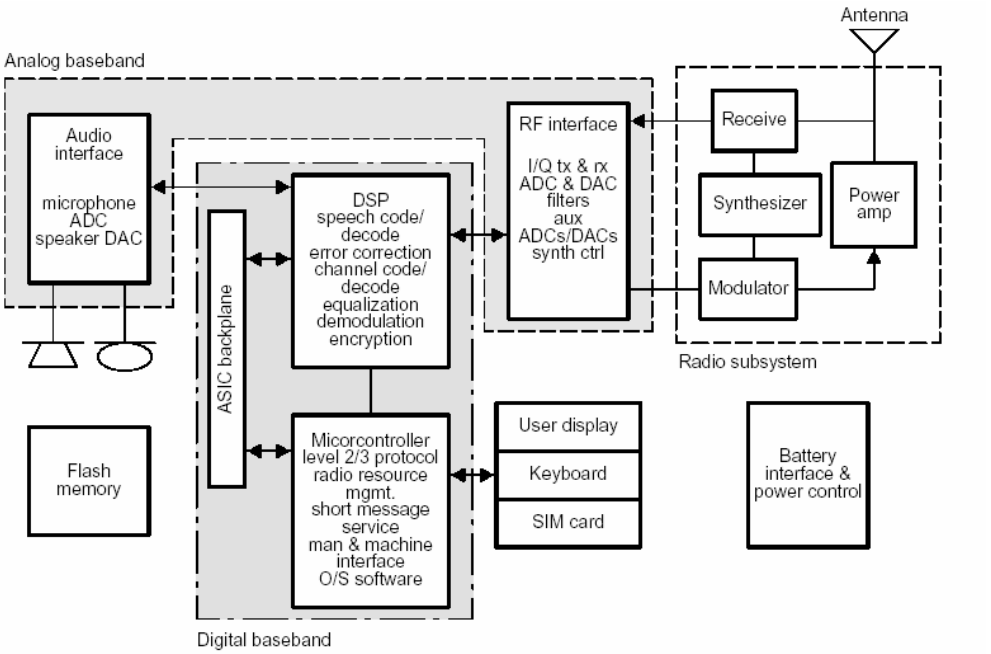

Figure 8: A cellular system of the second generation. A DSP handles all signal processing. A small, separate microcontroller, the user display, keyboard, and peripherals.

A large memory space is not used, and virtual memory, for example is not needed. Memory for operation and storage is allocated in one single device (there are no hard-disks).

The intelligence provided by the extra microcontroller is used to drive the few peripherals available, and the OS, but little else.

In an MP3 player, the block architecture is very similar, but the whole radio subsystem is not needed, as well as the microphone. An MP3 player can be embedded in a cellular telephone at

a minimal cost, and this is in fact what Nokia and other producers of cellular handsets have been doing in the last years. In the future much more functions will be integrated in the handsets, and this is already leading to advanced DSP architectures, as we review in the next section.

2.3 Modern DSP Architectures

2.3.1 The TMS320C55 series

Texas Instruments is the world leader in the DSP market. Other companies such as Lucent and Motorola have been also very active in this sector. I will look in this section to the architecture of some leading DSP processors, specially the TSM320C55 series from Texas Instruments, a widely used chip.

Fig. 9 shows a diagram of the architecture of the TMS320C55 CPU [TI 2000, TI 2004]. The CPU has a Harvard architecture with one read-bus for code, and three read-buses for data. Additionally, there are two write buses for data. Therefore, in the ideal case, this processor can read one instruction, read three operands, and write two results, all in the same cycle. The processor offers therefore extreme data-throughput, desirable and needed when processing streams of signals.

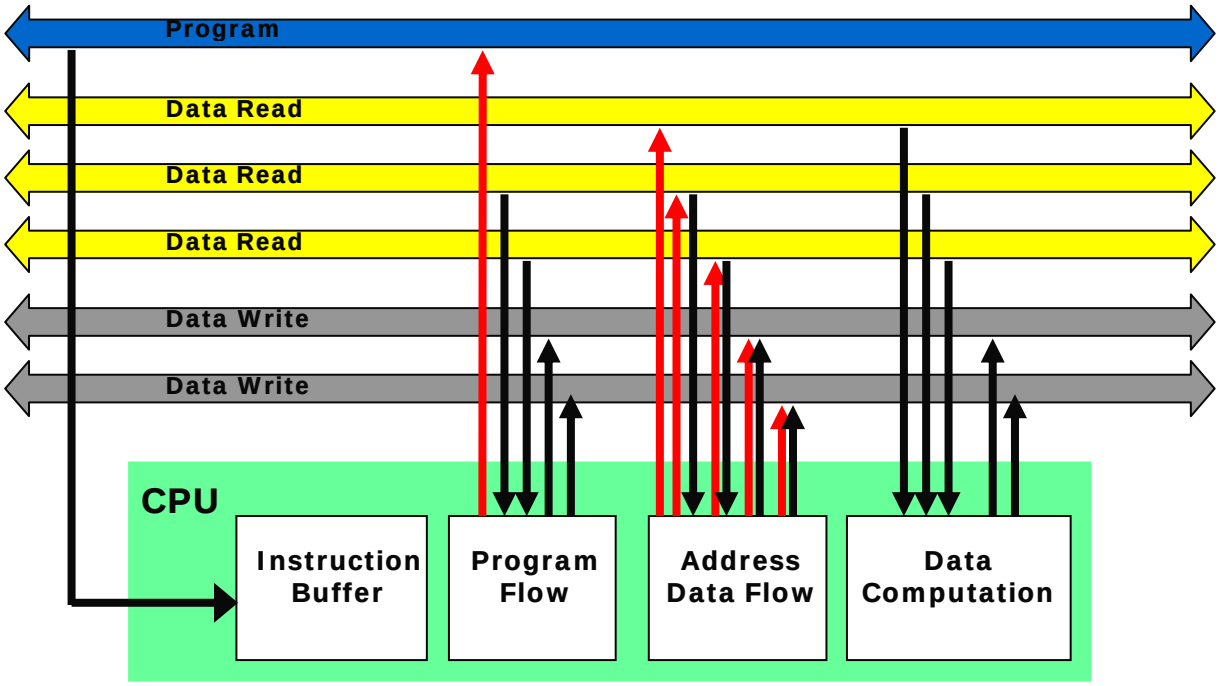

Figure 9: Architecture of the TMS320C55 DSP

A program flow unit (P) handles the sequencing of instructions, in close interaction with the address-dataflow unit (A). The A-unit can generate addresses of operands automatically, once data has been read or written. This greatly simplifies accessing arrays or sequential data stored in memory. The instruction buffer unit (I) buffers instructions, which can be of variable length [TI 2002], and provides them, decoded, to the P, the A, or the D unit. The I unit also stores code which can be used repetitively without having to access the external memory. A

single instruction or a block of instructions can be used repetitively. The data computation unit has multiple execution modules, as shown below.

Fig. 10 shows a diagram of the data computation unit. A barrel-shifter provides a shifting range of up to 32 bits (left and right). Such a barrel shifter is very useful for aligning byte or half-word data in a single cycle. The ALU performs addition, subtraction, and comparison with two 32 bit operands, or it can split into two ALUs with 16 bit operands each. The ALU can therefore execute two operations simultaneously. Since in signal data processing the resolution of the data rarely exceeds 16 bits, this split mode is very useful. Two MAC units complement the processor. Each MAC unit can multiply two 17-bit arguments, and add them to a 40 bit partial result in each cycle. All data is handled internally through the D unit registers.

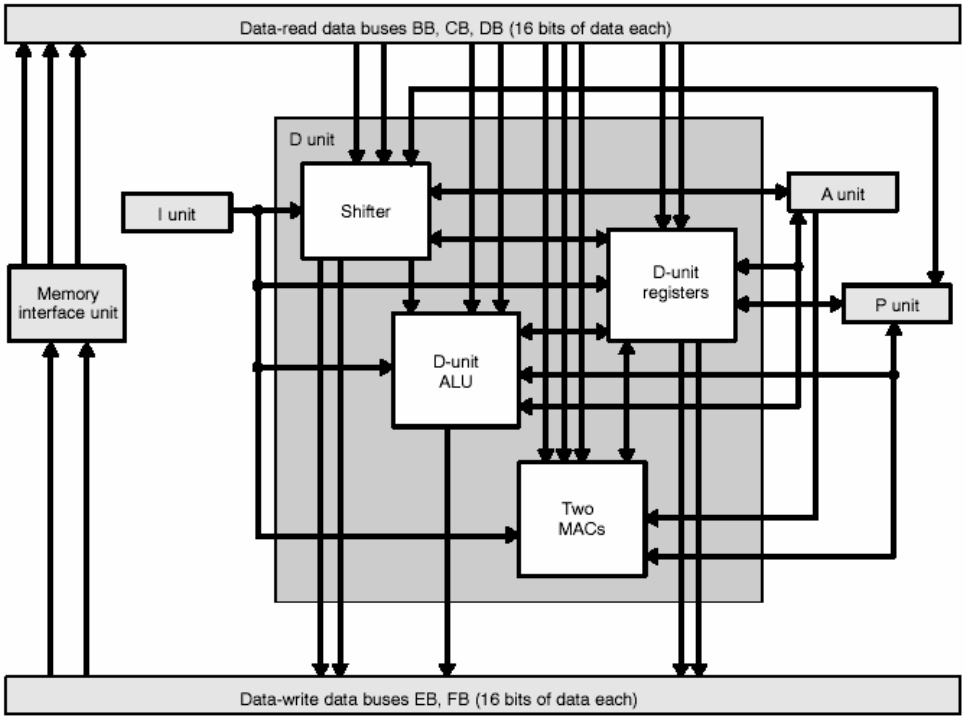

Figure 10: Architecture of the D unit of TMS320C55 [TI 2004]. The D unit includes two MAC units, one ALU, and a barrel shifter. The D unit registers can be accessed by the address and program flow units.

As the architecture of the TMS320C55 units show, this processor can be considered actually a VLIW (Very Long Instruction Word) CPU. The coding of the instructions is done using a variable format, and two or more instructions can be passed to the CPU in a single cycle. While the D unit is performing two multiply-accumulate operations, the address unit can be computing the addresses of the next operands, and the flow unit can be updating the PC to start a new iteration of a loop. In a pure-RISC processor, a single instruction of a TMS320C55 would have to be coded in seven or eight atomic RISC instructions.

Such high-density code is not useful or needed for general purpose processing. A compiler seldom access long arrays or needs to compute filters. Pointers and more variable branching patterns would put a DSP in trouble. But in signal processing, multiple instructions can be started more easily due to the more regular computation patterns used.

Another area in which a regular computation pattern helps, is in extending the depth of the pipeline. The TMS320C55 uses a very long pipeline, with many stages. First, fetching and decoding is done in four stages (two prefetch, one fetch, and one decode stage). Execution

adds eight additional stages to the pipeline. We have 12 pipeline stages in total, and therefore we can talk of the TMS320C55 DSP as of a superscalar unit. The eight execution stages are: Decode (D), Address (A), Access 1 (AC 1), Access 2 (AC 2), Read (R), Execute (X), Write (W), Write Memory (W+). Superpipelining is not very advantageous when exceptions disrupt the pipeline flow. The processor loses much time saving the CPU status and resetting. But in DSPs exceptions are less probable than in general purpose processors and can be avoided more easily. Number crunching while a stream is passing through the CPU is more important than handling exceptions, and thus virtual memory is not used, for example.

In addition, the TMS320C55 has programmable idle modes and automatic power saving features which bring down the processor frequency when the processor is not needed to perform at top speed.

2.3.2 The TigerSHARK Architecture

The TigerSHARK is a DSP architecture developed by the company Analog Devices [Fridman 2000]. It has a series of advanced features, which correspond to the kind of DSP chips becoming available in 2004 and the next few years. The main innovation is the use of “short vectors” in order to process information in SIMD (single instruction multiple memory) manner. Although the TigerSHARK has only a clock frequency of 150 MHz (for saving power), it can deliver 3.6 GigaOPS (operations per second) working on 16-bit data. At these enormous processing rate, graphic processing for 3D applications becomes much easier.

Fig. 11 shows a diagram of the TigerSHARK processor. It is a superscalar architecture: instructions which can run in parallel are identified at compile time and are started at the same time. The TigerSHARK can process data in fixed-point, but also in floating-point format. Even with floating-point, a multiply-accumulate operation is executed in a single cycle. A significant difference to the TI chip is the large number of registers visible to the programmer: 128 registers, divided in four orthogonal register banks (x, y, j, and k-registers). The buses connecting the computational units to memory (in a Harvard architecture) are very wide: 128 bits, which allow computation units to access four 32 arguments, or eight 16 bit arguments simultaneously. The J- and K-ALUs are used for generating memory addresses, but they also support general purpose arithmetic.

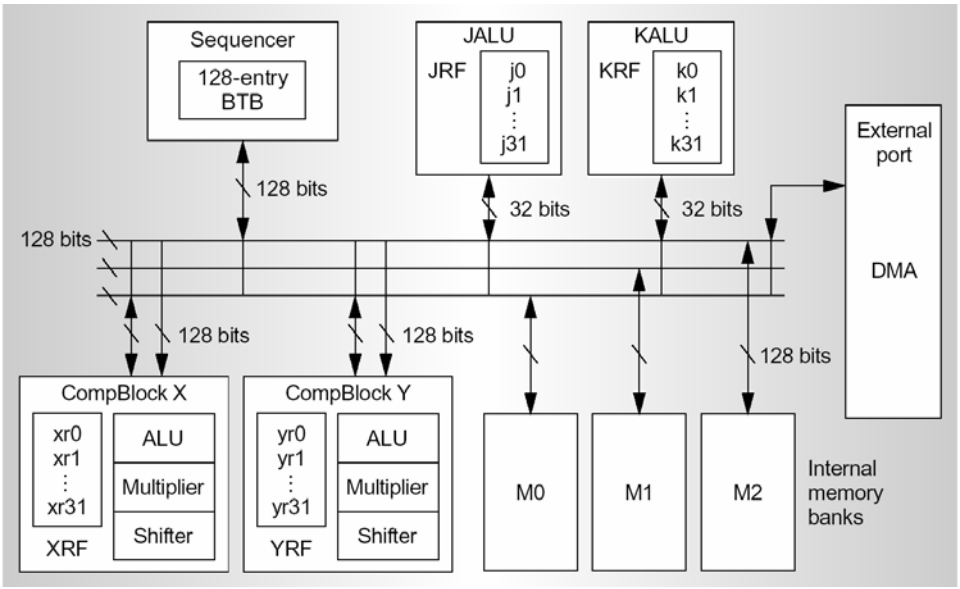

Figure 11: The architecture of the TigerSHARK DSP [Fridman 2000]. The data buses are 128 bits wide and can transport up to eight 16-bit operands each.

Each of the two computation blocks can perform two 32-bit multiplications, an addition, and also a subtraction per cycle. For 16-bit data, each unit can perform four multiplications, 4 additions, and 4 subtractions per cycle. This means that the two computational blocks can execute 24 operations per cycle, for a total of 3.6 GOPS at 150 MHz. The 16 data elements processed arrive through the two 128-bit long buses. They can be conceived as "short vectors" processed by the same instructions. There are several different ways in which the data can be processed in parallel using this processor, but the main message to get from this diagram is the progressive widening of the data-path and the effective use of vector operations. As we discuss later, parallel operations can save energy in a low-power processor.

3 Moving from voice to data centric systems

3.1 Hybrid Processors

The main advantage of DSPs is that they can be used as number crunchers, using limited frequency and power, due to the regularity of the computational load (mainly convolution operations for voice or signal processing). The relentless advances in chip fabrication are putting more and more computational power in each DSP. As the table below shows, the same size of DSP chip was delivering around 5 GIPS (Giga Instructions per Second) in 2000, compared to 5 MIPS in 1980. In the year 2010, 50 GIPS will become a possibility. One Megabyte of internal memory will be standard and the power pro MIPS will fall from 250 milliwatt per MIPS to 1 microwatt per MIPS. 50 million transistors will provide additional functionality.

Table 2. DSP integration through the years (typical device capabilities).

	1980	1990	2000	2010
Die size (mm)	50	50	50	5
Technology (micrometers)	3	0.8	0.1	0.02
MIPS	5	40	5,000	50,000
MHz	20	80	1,000	10,000
RAM (bytes)	256	2,000	32,000	1,000,000
Price (dollars)	150	15	5	0.15
Power (mW/MIPS)	250	12.5	0.1	0.001
Transistors	50,000	500,000	5 million	50 million
Wafer size (inches)	3	6	12	12

Figure 12: Past and projected evolution of DSPs [Gene 2000]

With all this computing power available, it is obvious that there will be a pressure to integrate more functionality in the classical cellular telephone, the killer application in the mobile marketplace. A single unit will have the computing power to be a telephone, a database, a general purpose computer, a gaming machine, a music player, or a digital camera. Therefore, the future of mobile devices consists in moving from audio-centric to data-centric architectures.

The architecture of a processor for signal and general purpose data processing can only be a superset of DSPs. Fig. 13 shows the proposed architecture for a next-generation Texas Instruments multimedia processor. A DSP core, which is essentially a TMS320C54 device, provides the signal processing capabilities. The analog/digital interface is directly managed by the DSP core. A low-power general purpose processor handles the general purpose operations. It is an ARM925 core, in charge of the operating system and management of peripherals [Brash 2002]. It can also overview the security and privacy of the device, one of the aspects which should be greatly improved for mobile devices in the future. The ARM core has been extended with such security features [York 2003].

The peripheral interfaces are varied: USB, serial, different classes of memory modules, or general I/O. At the center of the architecture sits the memory traffic controller. The processors can work on the same data, although the DSP will be handling streams, while the ARM core will be handling general code. Special interfaces provide DMA for a digital camera or even video, with a preview engine independent of the DSP core.

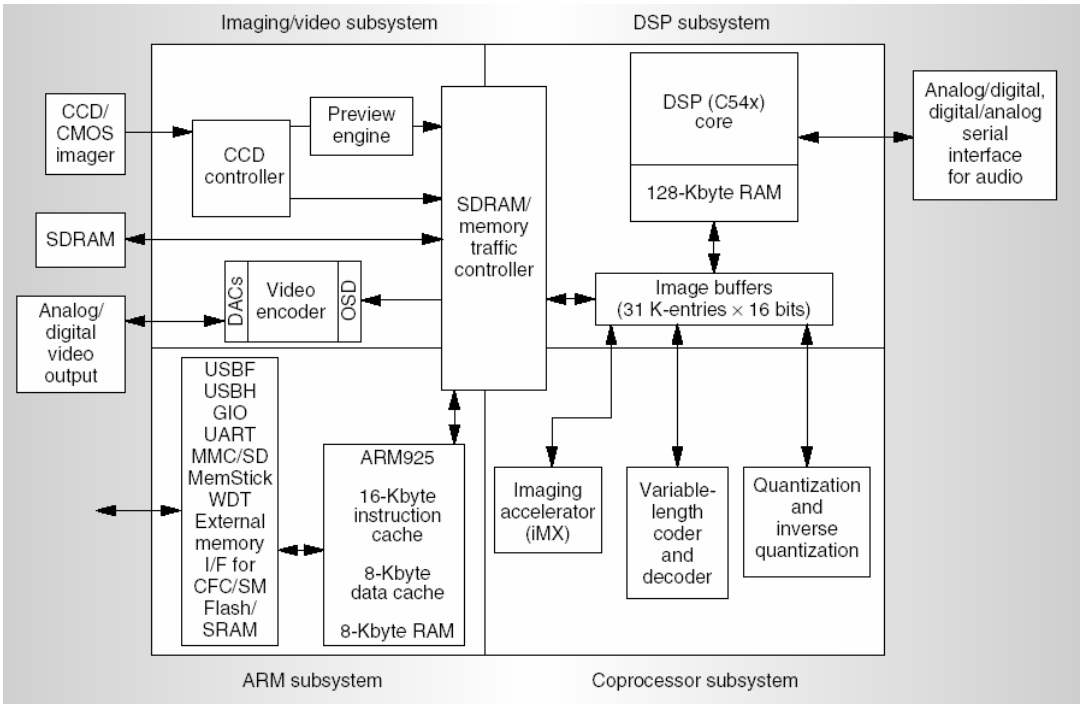

Figure 13: Architecture of a hybrid DSP and low-power general purpose core processor [Talla 2004]

Some circuits (ASICs) provide special functionality, like quantization, encoding/decoding, and image acceleration. A sufficiently fast DSP could handle all this functions, but once the processor can contain 50 million transistors, much more special units can be added. These special ASICs are just coprocessors for the whole system. As Fig. 13 shows, the complete multimedia system consists of four well-defined areas, communicating through memory. Anticipating such architectural developments, some companies, such as ARM Ltd, have enhanced their general purpose processing cores with DSP enhancements [Francis 2001].

3.2 Systems on a Chip (SOC)

The kind of design discussed above (similar also to the TriMedia Processor [Sijstermans 2001]) is very important for systems integrators, because the main required functionality has been reduced to a single chip. The motherboard for a complete mobile device has only to integrate this chip, memory, a display, some buttons and plugs and a wireless communication chip. The technological trend points toward Systems on a Chip (called SOC).

With SOCs, mobile device architecture becomes pretty simple, because the complexity is pushed from the motherboard to the multimedia chip. This is the same process we observe in the desktop market, where embedded PCs are now available at a low-cost. This has become only possible because most of the functionality of a PC can be concentrated into just a few chips.

The convergence of DSPs with general purpose processing cores leads to a fundamental problem: once a company achieves a lead in this new market (it could be Texas Instruments), the choice of operating system becomes crucial. An embedded OS offered together with a canonical multimedia processor would speed-up the design and time to market of mobile devices. Until now a few alternatives are available, but the cellular telephone market is still very different from the PDA market. Microsoft has been pushing the Windows CE OS as the solution for the mobile devices, but other companies, such as Symbian, are still dominant in the cellular market. And there is also embedded Linux, which has been also making inroads in the embedded device market. It is too soon to know which company will emerge as the winner, but network economies will probably lead to a de-facto monopoly in the mobile devices market, once the new generation of chips is widely available.

Another point to be mentioned here is the trend towards integration of DSP functionality in traditional RISC or desktop PC chips. Such enhanced RISC devices have been introduced to concentrate all processing in the desktop computer processor. It could be that, if a clear winner appears in the signal-data-processing chip arena, that this chip could be used as coprocessor for desktop devices.

4 Intelligent Power Management

4.1 Lowering Voltage

The two most important features of a mobile device are: a) real-time responsiveness, and b) low-power operation. A mobile device processing signals must never stop when reproducing audio, and must show images at a constant rate. Consumers are very sensitive to such issues. Real time responsiveness means that the mobile device must contain a number-cruncher, a DSP or an enhanced DSP hybrid chip. Low-power is also crucial. Battery life must be long so that the devices are truly mobile. A PDA which cannot be used in a long flight is useless. Fig. 14 shows the power consumption of a desktop processor for a 3D game application, decoding an MPEG movie, or a 3D game (in 2000). 24 to 22 Watts are nothing out of the ordinary, but impossible in a mobile device.

Faced with this power-consumption problem, designers of modern DSPs and embedded processors have come to a variety of solutions which can limit the energy used. On the one hand, transistors have become smaller and need less power to be operated, but we have more of them on a chip. On the other hand, voltage and operating frequency reductions can be used to save energy.

Lowering the frequency of a device is always an option. However, if the chip is made slower, this runs against the trend towards faster units with additional functionality. Other solutions are needed, or additional techniques are needed.

	1999 Power (Watts)		2000 Power (Watts)
	3D Game	MPEG 2 Movie	3D Game
CPU & L2 Cache	9.5	7.9	9.5
Memory Controller	1.2	0.9	1.6
System Memory	1.4	1.4	1.3
Graphics Subsystem	2.4	2.4	2.4
I/O Subsystem	0.5	0.6	0.6
Audio	1.5	1.5	1.6
Modem			0.4
Hard Drive	0.7	0.0	1.3
DVD Drive	1.4	3.0	1.4
1394 Controller	0.0	0.0	0.0
USB	0.0	0.0	0.0
CardBus	0.1	0.1	0.2
LAN	0.4	0.4	0.4
Power Supply	2.0	2.0	2.6
Charging	0.1	1.0	0.0
Cooling	0.5	0.5	0.5
Other	0.3	0.3	1.0
TOTAL	22.0	22.0	24.8

Figure 14: The Table shows the energy consumed by a desktop computer in 1999 and 200, for running a computer game or showing an MPEG movie.[Kolinski 2001]. Almost 10 watts are consumed by the CPU alone.

Lowering the voltage of the chip is one possible strategy, and it has been followed faithfully by the computer industry until now. Usually, processors for laptops operate at lower voltages than their desktop counterparts. There are now DSPs being operated at voltages near 1 volt, nearing the threshold at which CMOS transistors can still operate reliably [Forestier 2000, Hass 2001].

The reason for reducing the voltage of a processor is that the power consumed by the chip is proportional to the square of the voltage. Modern mobile systems can send the processor into sleep mode when the system is idling, waiting for input to a request. In sleep mode, parts of the chip can be disconnected or the voltage can be lowered. Lowering the voltage has the effect that the maximum clock frequency possible has to be reduced. The chip must be made slower in order to avoid data corruption in the internal data lines. However, this is not the optimal solution.

Consider a person watching a video with a mobile system. When the video is running, the full voltage is applied to the processor, even if each of the 30 frames per second can be processed in less than $1/30^{th}$ of a second, let us say, $1/60^{th}$ of a second. The processor is working at full speed, but effectively idling half of the time. The solution to this problem (since the voltage

cannot be lowered and raised between frames) is to use an intermediate voltage. If the full possible voltage is 3 Volts, the same processor running at let's say 1.5 Volts, could deliver the desired performance. This technique, voltage on-demand, can only be used if the mobile device is sophisticated enough to measure the actual load of the system and if the processor has been engineered to work with a variable voltage. This technique is called dynamic voltage scaling and is one of the most powerful ideas being incorporated today in mobile devices.

Until now, engineers have been very successful in lowering the power needs of DSPs. Fig. 15 shows a curve of what is called "Gene's Law". It is the equivalent of Moore's Law for power consumption. It states that the power needed per computational MIPS is halved every 18 months. This exponential reduction in the power required for processing will continue in the foreseeable future, and will lead to a reduction by about a factor ten of the power needed today for a MIPS when compared to the power needed in 2009.

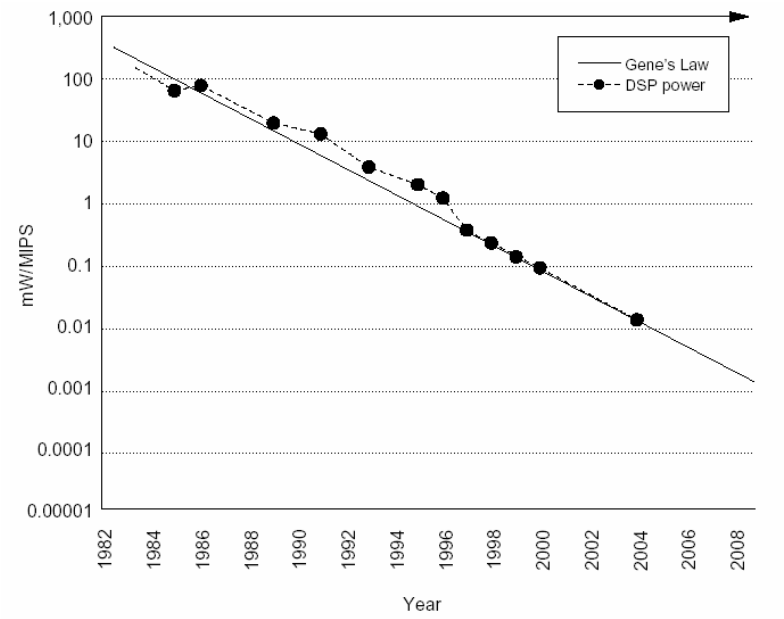

Figure 15: Gene's Law: The energy consumed per DSP-MIPS falls exponentially. The energy per MIPS halves every 18 months, that is, it is reduced by almost a factor ten every four years [Gene 2000].

Remember, however, that circuit integration is also doubling every 18 months, so that the power used by a DSP will be kept constant because the MIPS delivered also tend to double at the same rate. Additional solutions are needed in order to increase the battery life of mobile devices. One of them is parallelism.

4.2. Increased Parallelism

What would you do with 50 million transistors? You can add memory to your design, you can make the ALU faster, you can have more registers, etc. In a processor for the desktop all alternatives are interesting because, for all practical purposes, there is unlimited energy available, even if the processor is getting hotter than a cooking plate. In a mobile device power rapidly becomes the dominant issue, the one that limits the architectural choices.

Additional memory, for example. Mobile devices usually have static RAM, which is faster than dynamic RAM. But static RAM is made of CMOS transistors, millions of them. Most of the time, almost all the memory is just sitting there, keeping data available, and refreshing every bit. CMOS circuits are used to perform the switching. But CMOS circuits become more

and more inefficient when the size of the transistor elements is reduced. Leakage power becomes the dominant issue. We can put more and more transistors in the same area, but at an exponential increase in the cost of the power leak. From the point of view of the processing to energy trade-off, it is better to have as much of the chip as possible doing useful work. This calls for increased parallel operation, so that even memory is used more efficiently.

As hinted before, the power consumed by a CMOS processor is proportional to the product of the capacitance C, the square of the voltage V, and the frequency F, that is

$$\text{Power} \sim CV^2F$$

Lowering the voltage of the processor provides quadratic power savings. However, the maximal possible frequency is proportional to the voltage, so that the chip becomes slower [Miyoshi 2002]. But then the following can be done: assume that at voltage V the maximum possible frequency is F. Diving the voltage by half, reduces the frequency by half also. Total power is now one eight of the original, but the system is half so fast. The total savings per MIPS are a factor of four.

Adding a computational unit to the processor, which should be always as busy as possible, increases the capacitance C of the whole processor (without doubling it) but makes the processor, in the ideal case, twice as fast. The processor is as fast as it was before the voltage was reduced, and the savings are less than a factor of four, but near four. The exact factor depends on the efficiency of the parallel operations: power savings can now be achieved using clever compiling techniques. And if computer architecture history has taught us a lesson, it is this: it is easier to write faster compilers than build specialized hardware.

Modern DSPs can therefore become more effective if they do more work in parallel. Each computation unit runs with less heat, but we have now not one, but two or more of them.

Fig. 16 shows a design for dynamic voltage and frequency control [Flautner 2003]. A hardware performance monitor is measuring the desired performance requested by the applications, and turns down the voltage and clock of the whole system, if needed. It turns up as soon as the mobile device requests higher throughput. The operation of the IEM (intelligent energy management) module is invisible to the user, but saves much more power than the current schemes with sleep modes.

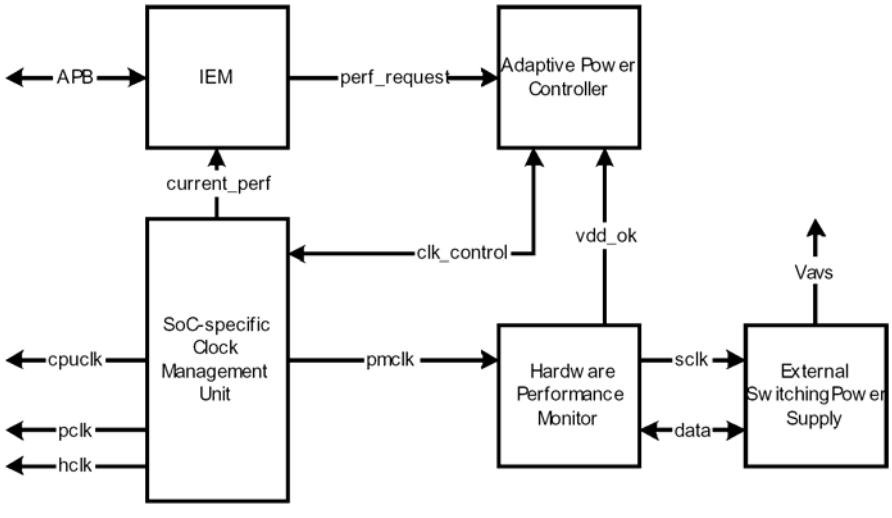

Figure 16: An intelligent energy management unit druving the clock and voltage of a microprocessor [Flautner 2003].

5 Hardware Architecture Performance and Power Management

We now come to the final section of this review, before extracting some consequences and trying to foresee future developments.

We have seen that DSPs have become so common in mobile devices because they provide the number crunching needed for real-time operation at low power costs. But we have also seen that future mobile devices will integrate more functions and will require more computational power. In this section we will review some of the techniques studied by some groups, most notably the Oxygen group at MIT. The general theme is the same: how to save power without reducing performance. As will be shown, the problem can be transferred from hardware to software, so tha the question becomes, how can the compiler help to predict the use of the hardware in such a way that we use less of it? Exploiting the synergy between hardware and software, as was done in the first stage of the RISC revolution, can help to reduce the power requirements of new processors.

5.1 Pipeline control

The first technique we review is pipeline exposure [Asanovic 2000]. In many RISC processors, a deep pipeline is used and the final stage of execution usually consists in writing the result to memory or to a register (register write-back). Measurements done by the Oxygen group led to the conclusion that almost 50% of register rewrites are unnecessary. In many cases a temporary value is computed in a register and it is immediately consumed in the next instruction. RISC processors, with it register based architecture require always a write-back. Accumulator architectures hold the final result always in an accumulator, which is available right after the execution stage of the pipeline.

Power can be saved if the forwarding registers in the RISC architecture are made visible to the software. When a register is used immediately after a result has been computed, traditional forwarding logic routes the result to the ALU input. Software can then just cancel the write-back operation if it is unnecessary. The software can also have a view of the forwarding registers and can manage them in an intelligent way.

Fig. 17 shows a diagram of a RISC architecture where the forwarding registers have been made visible to the programmer (RS, RT, SD registers). The compiler can then decide that a register does not have to be written back, it suffices to pass it to register RS or RT as argument for the next operation.

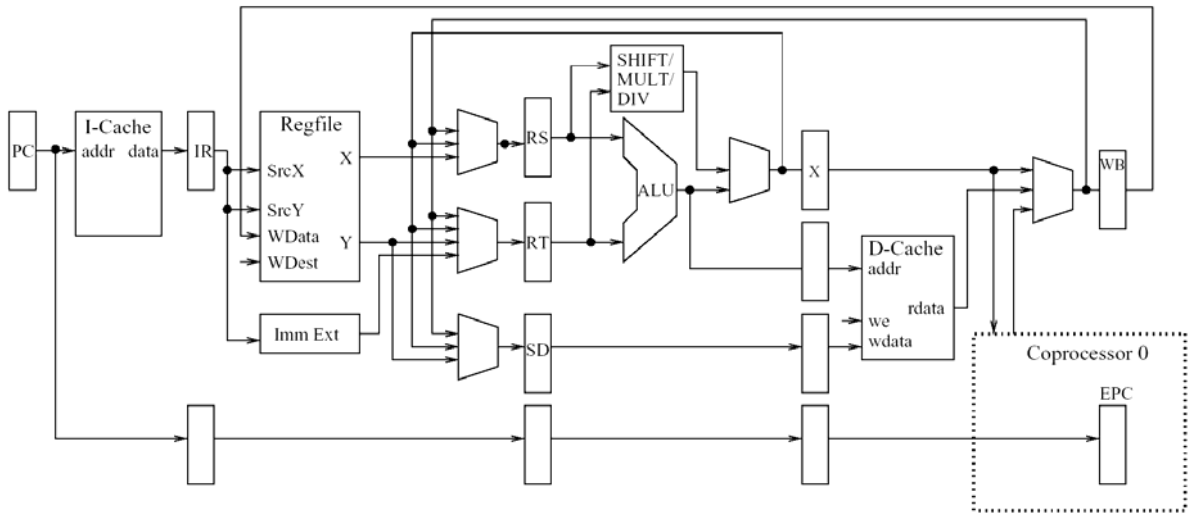

Figure 17:A pipeline with exposed forwarding registers [Hampton 2001]

Eliminating the write-back phase of arithmetical operations saves around 7% of the power used in a processor. It can seem that this is a small number, but remember that the savings is achieved without performance penalty and the additional hardware needed is almost trivial. The savings computed in simulations already consider the additional hardware needed. Fig. 18 shows an example. The second column is lower because less power was spent computing. Most of the savings come from the write-back phase, where almost 50% of the write-backs have been avoided.

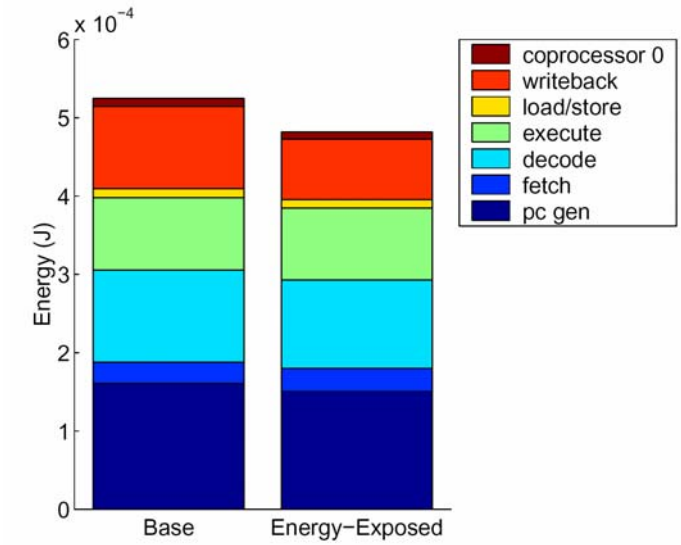

Figure 18: Energy savings (right bar) for a exposed pipeline architecture [Hampton 2001]

Exposing part of the pipeline to the compiler represents a problem when considering exceptions. When a exception occurs the pipeline has to be flushed and in-order-execution has to enforced. If a register has not been updated by a write-back, an error could occur. Saving the forward registers represents additional hardware and it could be that the savings from avoiding write-backs are lost updating processor status flags. The solution to this problem consists in saving the CPU status only at specific checkpoints in the code. If ten instructions can be repeated without affecting the computation, then a single checkpoint at the beginning of the sequence can be defined. Processor status is only saved there and any exception within the block restarts the whole block. Processors usually have a block length of only one instruction. The idea here is to make the block longer.

In general purpose computing this technique could be used, in mobile devices it is probably more useful, since the amount of peripherals and exceptions is limited. One source of exceptions, for example, is virtual memory, which is not used in mobile devices. Peripherals are usually not attended by a DSP. The length of basic blocks in which exceptions can restart the block is longer in such mobile devices. Pipeline exposed compilation techniques are therefore feasible and could become standard in future microprocessors, mobile or not.

5.2 Cache hit prediction

Desktop processor employ large caches. Mobile systems have smaller caches, if at all, but caches will certainly become more and more important because the high throughput of VLIW architectures is only reachable using a fast memory system. Tag checking, as done in caches, is an expensive operation in terms of power. If tag checking can be avoided, power can be saved [Witchel 2001].

In many cases the compiler knows the structure of the data being accessed. If the compiler knows what kind of cache is being used, and its size, then it can determine in many cases that a tag check is not necessary and a direct memory read is done. The compiler must make sure that no error is possible.

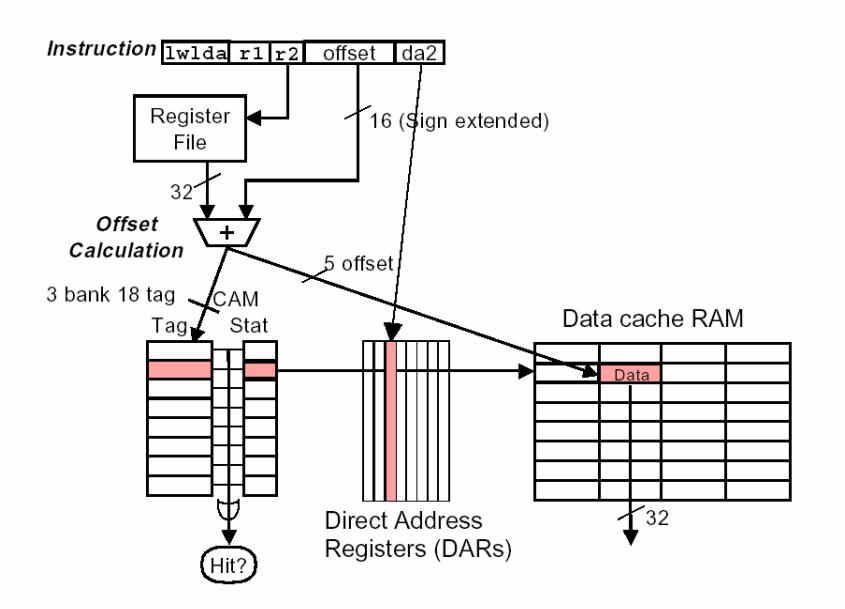

Figure 19: Directly addressable cache diagram [Witchel 2001]

Tag checks are so energetically expensive, because they are performed in parallel for every block in a set within an associative cache. Fig. 19 shows that once the block in a cache is known, the tag check can be avoided by addressing the cache directly through special registers.

5.3 Variable instruction length

The improvements mentioned above lead towards more complex instructions, which is some sense runs against to the traditional wisdom that fixed-word encoding is easier to decode and to pipeline, and therefore more efficient.

Modern DSPs with its VLIW style of architecture have exacerbated the memory throughput problem. In many cases a VLIW consists of many repeated portions or many zeroes, and it is therefore natural to ask if the long words (of even variable size) can be efficiently compressed to fit into memory and decompressed after fetching them, on-the-fly, so that decoding is not affected. This technique has been studied at MIT [Pan 2001]. Figure 20 show the findings of this study. Code can be compressed up to 75% of its original size, without loss in performance.

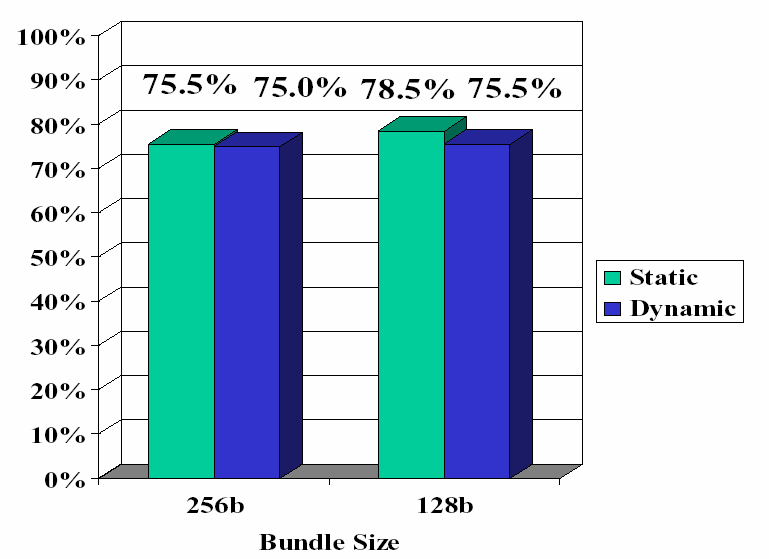

Figure 20: Code compression possible for a processor [Pan 2001]

ARM Ltd. has experimented with a compressed subset of the instruction set. The most used instructions are encoded in 16 bits, and so more instructions can be read by the processor. These so-called “Thumb” extensions can increase the memory throughput for code up to 30% in typical programs [Phelan 2003].

5.4 Data compression for transmission

In digital cellular telephones, the voice signal is compressed before it is transmitted. This is precisely one of the main advantages of digital over analog telephony. If this is done for audio, it could be also done for data.

It has been calculated that in modern PDAs, sending a bit over a wireless link costs 1000 times more power than one single arithmetic operation in the processor. This means, that if the data can be compressed and decompressed on the receiving side, so that one transmitted bit is saved using less than 1000 arithmetical operations, then a net energy saving can be achieved. Since, as we saw before, the power per MIPS for DSPs is constantly falling, then even more sophisticated encoding/decoding techniques could be used in the future.

Web browsing is a good example. If text is encoded before being transmitted, significant power savings can be achieved. Images are already sent encoded (for example in JPEG format) so that less saving are possible here, but dictionary based methods could become feasible, if it is possible to standardize such procedures.

Figure 21 shows the results of a study conducted at MIT, in which different encoding-decoding techniques were tested in a mobile device [Barr 2003]. The total energy used is measured in Joules and the last bar shows the cost of sending the uncompressed data. The other bars show the energy cost when different encodings are used. The savings can be significant. On average, and over a set of different benchmarks total savings of 7% seem achievable. Higher savings will become possible with future generations of DSPs. Also, note that the encoding and decoding form a pair, in which computational load can be shifted from one side to the other. There is an asymmetry there. Depending on which side is the mobile device, the computational load can be transferred to one side or the other. A non-mobile receiving unit can spend more energy decoding and save energy to the mobile device. A cooperating wireless environment can help to increase battery life.

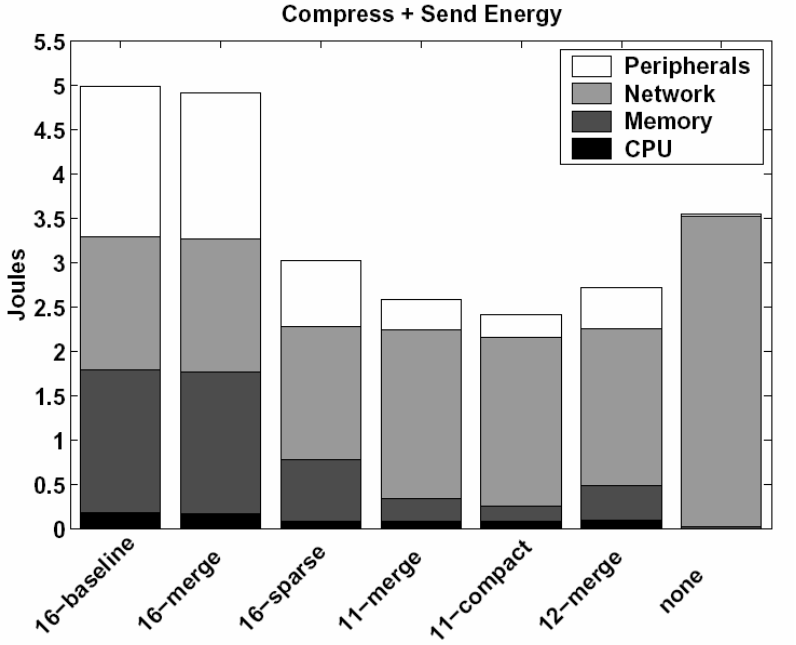

Figure 21: Total energy used for compressing and sending a file [Barr 2003]. The last column to the right shows the energy used without compression. The other bars show the energy used with different compression schemes. The different shadings show the energy consumed by the peripherals, network, memory, and CPU. Without compression, network energy dominates. With compression the network energy is lower, but some CPU energy must be added.

6 Conclusions and Outlook

I can summarize this review of mobile devices architectures with a list of the main ideas discussed in the text:

- Most of the computing power available now in the world is already provided by embedded and mobile devices. Just like most of the DNA in this planet is carried by bacteria and insects, most of the computational intelligence is now available not in mainframes or PCs, but in everyday objects. It has been projected that until 2010, up to 30000 million devices will be connected to the global data network [Clark 2002]. Most of them will be embedded systems.
- Mobile devices must operate in real time and consume low power.
- Real time operation implies number crunching ability for digitized signals.

- Cellular telephones are currently the “killer application” in the realm of mobile devices, with around 1500 million cellular subscribers in the world. However, there are many other kinds of mobile devices being built now and their variety is increasing every day.
- The architecture of mobile devices has been, until now, voice-centric. Special processors, DSPs, are used to process audio data in real time, providing the encoding/decoding and processing capabilities needed.
- DSPs are specialized processors which are so fast because they perform operations specially useful for digitized signals.
- Moore’s law is increasing the available computational power exponentially. Mobile devices of the next generation are evolving from a voice-centric to a data-centric architecture. The cellular telephone is evolving to be a more general device.
- Processors for future mobile devices must still work with low-power, but will incorporate a DSP core and a general-purpose core, plus additional special circuits, in a single package.
- A dominant embedded and mobile devices operating system has still to emerge, for the new generation of mobile devices.
- Future mobile processors will incorporate more parallelism in superscalar designs. VLIW and SIMD architectures will become more popular, because they allow to reduce the frequency and voltage of the processor chips without losing performance.
- The software-hardware synergy has still to bee fully exploited for mobile devices [Wolfe 2004]. There are many ways of reducing the energy used by the processor, which rely just on the availability of more intelligent compilers.
- Current research in mobile devices architectures for low power devices has concentrated in exposing the data pipeline to the software, handling efficiently compressed variable length code, reducing the power consumption of caches and memory, and also in exploring extreme data compression for saving transmission power.

In the future all these trends will lead to more efficient and powerful embedded and mobile devices. It is in this sense that we can start thinking about not just embedded systems, but “embedded intelligence”.

Embedded intelligence means that everyday objects can use their tremendous computing power in order to assist the user, anticipating even his or her commands. A mobile device which learns to automatically download the newspaper read by the user, caching the sections the user usually access, which learns to activate household items in the morning, which can help to find the car parked on the parking lot of a mall, which can provide GSP guided directions to any address, which can be a telephone and music player, as well as video camera, will be much more useful. There will be room for truly specialized devices (the “dumb” telephone, for example, will never die), but the future belongs to integrated mobile devices with plenty of computational intelligence.

Regarding connectivity, the mobile device of the future will be always on and always connected to the network There is still much work to do regarding packet switching standards for mobile devices [Krashinsky 2002], but if the mobile devices of the future are to be as intelligent as mentioned above, they must be seamlessly and permanently in contact with the global data network, just as desktop PCs are now. The big challenge is to still keep the power usage at a low level. One possibility which has been studied, is to embed many small processors in our office and city environment, which can assist the mobile devices and reduce their power needs. A user sitting in a taxi cab, for example, can then receive packets and route

them through the computer in the taxi, which has a more powerful antenna and server. The mobile device is then "cyber foraging" the computational resources of the environment [Balan 2004]..

The future will show which of these scenarios become reality, but it is already true that mobile and embedded devices are the wave of the future in computing, and that academic computer architecture research has gained new impetus and urgency derived from the needs of low-power computer architectures.

References

[Addra 1999] M. Addra, D. Castel, J. Dulongpont, P. Genest, "Microelectronics in Mobile Communications: A Key Enabler", *IEEE Micro*, September-October, 1999.

[Asanovic] K. Asanovic, "Energy-Exposed Instruction Set Architectures", *Work in Progress Session ,HPCA-6*, Toulouse, France, January 2000.

[Barr 2003] K. Barr, K. Asanovic, "Energy Aware Lossless Data Compression", *The First International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, May 2003.

[Balan 2004] R.J. Balan, "Powerful Change Part 2: Reducing the Power Demands of Mobile Devices", *Pervasive Computing*, April-June 2004, pp. 71-73.

[Brash 2002] D. Brash, "The ARM Architecture Version 6", ", White Paper, ARM Ltd., January 2002.

[Clark 2002] D. Clark, "Marketplace", *IEEE Internet Computing*, January-February 2002, p. 11.

[Flautner 2003] K. Flautner, D. Flynn, "A Combined Hardware-Software Approach for Low-Power SoCs:Applying Adaptive Voltage Scaling and Intelligent Energy Management Software", *DesignCon 2003*, ARM Ltd., 2003.

[Forestier 2000] A. Forestier, M. Stan, "Limits to Voltage Scaling from the Low Power Perspective", *Proceedings of the 13th Symposium on Integrated Circuits and Systems Desi*gn (SBCCI.00) , 2000.

[Francis 2001] H. Francis, "ARM DSP-Enhanced Extensions", White Paper, ARM Ltd, May 2001.

[Frantz 2000] G. Frantz, "Digital Signal Processor Trends", *IEEE Micro*, November-December 2000.

[Fridman 2000] J. Fridman, Z. Greenfield, "The TigerSHARK DSP Architecture, *IEEE Micro*, January-February 2000, pp. 66-76.

[Hass 2001] K.J. Hass, J. Venbrux, P. Bhatia, **"**Logic Design Considerations for 0.5-Volt CMOS", *Proceedings of the 2001 Conference on Advanced Research in VLSI* (ARVLSI 2001) .

[Hampton 2001] M. Hampton, *Exposing Datapath Elements to Reduce Energy Consumption*, M.Sc. Thesis, MIT, June 2001.

[Hennessy 1999] J. Hennessy, "The future of systems research", *Computer*, IEEE, August 1999.

[Intel 2000] Intel Corp. *Intel Xscale core developers manual*, December 2000.

[Intel 2003] Intel, "Intel Mobile Application Architecture Guide", 2003.

[Kolinski 2001] J. Kolinski, R. Chary, A. Henroid, and B. Press, *Building the Power-Efficient PC. A Developer's Guide to ACPI Power Management*, Intel Press, 2001.

[Koushanfar 2000] F. Koushanfar, V. Prabhu, M. Potkonjak, J. Rabaey, "Processors for Mobile Applications", *Proceedings of the 2000 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, 2000.

[Krashinsky 2002] Krashinsky, H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown", *MOBICOM 02*, Atlanta, September 2002.

[Müller 1992] M.Müller, "Power efficiency & low cost: The ARM6 family", *Hot Chips IV*, August 1992.

[Miyoshi 2002] A. Miyoshi, C. Lefurgy, E. V. Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: Understanding the runtime effects of frequency scaling", *International Conference on Supercomputing*, June 2002.

[Pan 2001] H. Pan, K. Asanovic, "Heads and Tails: A Variable Length Instruction Format Supporting Parallel Fetch and Decode", *CASES 01*, Atlanta, November 2001.

[Patterson 2003] D. Patterson, J. Hennessy, *Computer Architecture, A Quantitative Approach*, Morgan-Kaufmann, 3rd Edition, 2002.

[Phelan 2003] R. Phelan, "Improving ARM Code Density and Performance. New Thumb Extensions to the ARM Architecture", White Paper, ARM Ltd., June 2003.

[Sijstermans 2001] F. Sijstermans, "The TriMedia processor: the price-performance challenge for media processing", *2001 IEEE International Conference on Multimedia*, 2001.

[Talla 2004] D. Talla, C. Hung, R. Talluri, F. Brill, D. Smith, D. Brier, B. Xiong, D. Huynh, "Anatomy of a Portable Digital Media Processor", *IEEE Micro*, March-April 2004, pp. 32-39.

[Tanenbaum 1998], A.S. Tanenbaum, *Structured Computer Organization*, Prentice Hall, 1998.

[TI 2000] Texas Instruments, „TMS320C55xE DSP, Functional Overview", June, 2000.

[TI 2002] Texas Instruments, "TMS320C55x DSP Mnemonic Instruction Set Reference Guide", October 2002.

[TI 2004] Texas Instruments, "TMS320C55x DSP CPU Reference Guide", February 2004.

[Turley 2002] J. Turley, "Embedded Processors", *Extreme Tech*, January, 2002.

[York 2003] R. York, "A New Foundation for CPU Systems Security**.** *Security Extensions to the ARM Architecture*", White Paper, ARM Ltd, May 2003.

[Witchel 2001] E. Witchel, S. Larsen, C.S. Ananian, and K. Asanovic, *34th Annual International Symposium on Microarchitecture ,MICRO-34* , Austin, Texas, December 2001.

[Wolf 2004] W. Wolf, "Embedded is the New Paradigm", *Computer*, March 2004, pp. 99-101.

[Wolfe 2004] M. Wolfe, "Supercompilers, the AMD Opteron, and your Cell Phone"**,** *Proceedings of the 18th International Parallel and Distributed Processing Symposium* (IPDPS'04), 2004.