

Embedded Systems Computer Architecture

(Extended Abstract)
Jakob Engblom

Abstract—Embedded systems are computer systems used as components in other systems. It is a very broad field encompassing a large number of very different requirements, and the computer architecture of embedded systems reflects this variation by a large degree of specialization to application areas.

Index Terms— Embedded Systems, Computer Architecture

I. INTRODUCTION

Embedded systems are computer systems used as components in other systems. It is a very broad field encompassing a large number of very different requirements, and the computer architecture of embedded systems reflects this variation.

By their nature, embedded systems are *special-purpose systems*. In general, compared to a desktop or server machine, the computer employed in an embedded system will address a rather narrow, well-known, and fixed application. This makes it possible to specialize the computer architecture to address this particular application.

The performance demands of the system are usually well-defined at the design stage, and they are not likely to change over the lifetime of the system. This means that the performance and capabilities of an embedded system are targeted to the needs of the application. Extra performance or extra features over and above the particular needs of the application are a waste, not a feature. Sufficient performance is indeed sufficient. This is why billions of old 8-bit processors are still sold every year into the embedded market – for many tasks, they offer an acceptable solution.

There are three factors that need to be balanced to determine the perfect computer base for an embedded system. The *performance* has to be sufficient. The *cost* has to be minimized. The *power consumption* (and heat production) has to be within design bounds. It is hard to satisfy all three goals simultaneously. Low power and low price also means low performance. Higher performance usually brings with it higher power consumption. Getting high performance cheap is always difficult. Sometimes, systems will need to be redesigned or specifications changed to accommodate the available processing power. Your mobile phone cannot currently have 3D graphics that can compete with a desktop

PC, due to power and size constraints.

The best balance of power, performance, and cost is usually found in computer architectures specialized towards a certain task. Barring a huge difference in production volumes, a general-purpose machine will always cost more and use more power than a special-purpose machine for the same problem. This is what gives the embedded systems space its unique diversity and room for innovation.

One consequence of the attractiveness of specialized processing is that a system will often have multiple processors, each specializing in a particular type of task. A common distinction is between two styles of processing: *control plane* and *data plane*. The control plane is the part of a system that makes decisions and controls its behavior; it is usually dominated by decision making and data lookup. For example, in a telephone switch, this includes setting up the circuit for a phone call. The data plane, on the other hand, is the part of the system that is in charge of processing and shuffling data around; it is dominated by repetitive data movement and computations. In the phone switch example, this is the part that actually transports the sound stream from sender to receiver. Once the control plane has set up a call, the data plane will take over and do the work as long as the call is connected. This model, splitting control decisions and data is quite common, even though it obviously does not cover all embedded systems.

A final property of embedded systems that is often overlooked is the longevity of the systems. Many embedded systems, especially in the military and aerospace fields, have very long lifetimes, often reaching into decades. This makes future parts and tools availability a big issue in the design phase, as longevity has to be planned for.

II. EXAMPLES

To give an idea for the wide span of systems that can be called embedded, we will go through some examples.

An advanced toy like the Lego Mindstorms robotics construction kit contains a fairly simple processor: an 8-bit Hitachi H8 processor with 32k of ROM and 32k of RAM provides the brains for this quite sophisticated system. This offers a cheap and effective solution for creating a very fun smart toy.

A typical (non-smartphone) GSM phone contains a number of processors. An 8-bit processor might take care of the user interface, games, etc., while a 16-bit DSP provides the processing power necessary for digital voice encoding and decoding. Apart from these main processors, the Bluetooth

Manuscript received February 10, 2004.

Jakob Engblom is a Business Development Manager at Virtutech (<http://www.virtutech.com>) and an adjunct professor at Uppsala University, <http://user.it.uu.se/~jakob>. (e-mail: jakob@virtutech.com).

unit in the phone contains an embedded 32-bit RISC processor used to process the communications protocol, as does the IR port. For smartphones that integrate more functions, 32-bit main processors are becoming necessary. So what we have is a small portable multiprocessor system.

Modern cars from manufacturers like Volvo, BMW, or Mercedes contain up to a hundred embedded processors. Some are powerful 32-bit processors used in engine control and similar compute-intensive tasks, while most are simpler 16-bit and 8-bit processors used to control various functions around the car (windows, locks, ACC, ABS brakes, etc.). The processors communicate with each other using buses like CAN, LIN, and FlexRay. Cars are extremely heterogeneous distributed systems. Since cost control is of essence for mass-production items like cars, each electronic unit is cost-minimized, even at the expense of a somewhat higher software development cost (since development costs are one-time expenses, while per-unit costs are incurred for each car produced). The processing power in cars is located where things need to happen; centralization is not an option.

Telephone systems contain a large number of embedded systems with varying computational needs and styles. For example, mobile phone base stations are computation-intensive digital signal processing systems. Such systems contain huge numbers of 32-bit or floating-point DSP processors to encode and decode radio signals and maintain the connections to the mobile phones. It is a very parallelizable system: each active phone requires the same processing of independent data, giving thousands of independent computation threads to spread across the DSP processors. They offer an almost perfect parallel workload. Several startups have tried to address this market with heavily parallel architectures.

Enterprise networking equipment like switches, routers and storage controllers make up another class of embedded systems. They are often quite similar to regular computers, containing one or a few 32-bit or 64-bit RISC processors, and running an operating system like Linux. However, the architecture is optimized to the movement of large amounts of data through the machine, using special line cards to take care of moving data while the main processor is only rarely involved. For the highest-capacity systems, custom CPU architectures are often employed, since regular processors are not well-suited to the task of packet processing.

Large military systems like radar stations and combat ships require enormous amounts of processing power, and here one can find regular multiprocessor servers working as embedded systems; albeit in special military-hardened cases. Even a Sun server can be considered an embedded system in the right circumstances! Often, military systems have tight space requirements, as ever more computing power is retrofitted to designs intended for far fewer computers.

Space-based systems offer another extreme: they need to employ special radiation-hardened cores and seldom enjoy the luxury of high clock frequencies or 32-bit processors.

III. THE MARKET

Embedded processors make up about 98% of all processors shipped (by number). Of about eight billion processors manufactured each year, only around 200 million find their way into desktops and servers. Looking at the overall semiconductor market, processors only make up about 2% of the numbers of parts sold, but about 30% of the revenue. So processors are clearly the highest margin part of the market to be in.

In the processor market, while 4-bit and 8-bit processors make up about 70% of the numbers, they only contribute a tiny fraction of the money. 32-bit and 64-bit processors (embedded or not) get more than 65% of the overall processor revenue, even though they are less than 10% of the numbers. Within the 32/64-bit category, desktops and servers takes almost all the money (50% of all processor revenue), thanks to the much higher price of server and desktop processors (usually hundreds of dollars apiece) compared to embedded processors (maybe tens of dollars, often less) [1].

So while embedded processors are dominant by numbers, we can see that it is very different on the revenue side. But 32-bit embedded processors are still a healthy and rapidly growing market that keeps attracting newcomers. ARM (www.arm.com), today's most common 32-bit architecture, is produced in about 600 million units per year.

ARM is a good example of a business model peculiar to the embedded world, the licensable processor house. ARM designs processor cores and licenses them to other companies who then create products containing the ARM cores (combining the core with various devices and memories to create sellable chip); ARM does not produce any chips of its own. This business model is also used by MIPS (www.mips.com), ARC (www.arc.com), Tensilica (www.tensilica.com), and others.

IV. PRODUCT CATEGORIES

The embedded processor market defines itself around a number of product categories, vaguely defined and featuring extensive overlap, but nevertheless they help organize and understand the market place.

A. Microprocessors

The classic microprocessor is a chip that contains just a processor, nowadays usually with integrated caches and sometimes memory controller. This is your SPARC, Pentium, and PowerPC processor found in regular office computers. Standalone processors are sometimes found in embedded systems, especially when lots of processing power is needed.

B. Microcontrollers

A *microcontroller* is the traditional embedded processing part. It encompasses not only the processor core, but also a number of peripheral devices like timers, serial ports, A/D converters, and network interfaces, along with some amount of program and data memory. The goal is to reduce the number of external chips needed, in order to minimize cost. Most microcontrollers are based on 8-bit or 16-bit processing

cores, with a few kilobytes of data RAM and up to half a megabyte of ROM or FLASH memory for code. Typical microcontrollers are the Atmel AVR (www.atmel.com) and Microchip PIC (www.microchip.com) families.

C. ASIP/ASSP

Recently, the term *application-specific instruction set processors (ASIP)* or *application-specific standard parts (ASSP)* have come into use to denote “super-microcontrollers”. These chips take the level of integration on a single chip to new heights, based on the enormous number of transistors available in 130nm or smaller silicon processes. They are also “application-specific” in the sense that they are quite narrowly targeted to particular applications, and aim to replace the traditional development of custom hardware. Multiple cores and 32-bit processors are common in ASIP/ASSPs.

One good example are the Texas Instruments (www.ti.com) OMAP chips, which integrate an ARM core with a DSP core, memories, LCD and keyboard drivers, and other devices to put most of the logic of a mobile phone onto a single chip.

Infineon (www.infineon.com) has a family of chips built around the C167 core that are very popular in automotive applications. The C167 chips feature multiple on-chip CAN controllers, waveform generators, A/D and D/A converters, and many advanced timers.

Sometimes, ASIP/ASSP chips are billed as *System-on-a-Chip* (SoC) solutions. The term “SoC” has been getting very popular in recent years to denote highly-integrated chips that encompass all parts of a complete system, especially in the context of ASIC design (see below).

One particular class of ASIP/ASSPs are the *network processing units (NPU)*. NPUs are chips designed specifically for networking applications, and include both control-plane and data-plane components. IBM’s (www.ibm.com) NP chips and Intel’s (www.intel.com) IPX are quite typical, combining a standard processor core with a large number of semi-programmable data pumps.

D. ASIC

Application-Specific Integrated Circuits (ASICs), are the ultimate embedded processing devices. ASICs are fully custom chips designed by the end user for the needs of a particular application. They can contain control logic, processing cores, memories, devices, buses, and whatever else an application might need.

Designing an ASIC is a very expensive process, and you will need large volumes or very special needs for it to be a viable proposition. As the number of gates that can be fit on a chip increases, ASIC design becomes ever more complex. The economics of ASICs are like classic printing: you have a high fixed setup cost to create the set of *masks* used to print the ASICs, but once the setup is done, the cost per unit is small. The more units you create, the lower the per-item cost will be.

One way to speed the design is to buy complex components from *intellectual property (IP)* providers. Most standard parts of an ASIC can be bought, from the simplest serial ports to

processor cores. Processor cores (discussed briefly above) are the largest part of the IP business, since processors are the most complex part to design in-house (not to mention the need to create support tools like assemblers and compilers).

In some cases, ASICs include full-blown home-made special-purpose processors. A classic example is Ericsson’s (www.ericsson.com) APZ processor, used in the AXE series of digital phone switches. It is a very specialized architecture designed for the single application of phone switches. Another example is Cisco’s (www.cisco.com) Toaster series of switch processors; standard processors cannot process packets fast enough for high-end parts, so a special architecture was needed. None of these are available outside the respective companies, making them ASICs and not microprocessors.

E. FPGAs

Field-Programmable Gate Arrays, FPGAs, are “soft hardware”. They are hardware chips whose function can be change by updating the contents of configuration memories. FPGAs are built from cells, small units implementing a small piece of logic controlled by configuration data. The cells are connected with a programmable interconnect to form complete circuits.

Compared to ASICs, FPGAs implement the same function in a much less efficient manner. Due to the obvious overhead in the implementation, FPGAs clock lower, exhibit higher power consumption, and contain fewer available logic gates. They also cost more per unit. But they are *reprogrammable*, and there is no setup cost like ASICs. This makes FPGAs more flexible and cheaper to design and work with.

FPGAs have always been popular as prototyping and validation tools: hardware designs can be validated by creating an FPGA and testing it, which is much faster and cheaper compared to creating a version of an ASIC.

In recent years, as costs for ASICs have increased drastically (startup costs are hitting millions of dollars for 130 nm and 90nm processes), FPGAs have become an alternative to ASICs in production units, especially for low-volume products. Currently, experts estimate that at volumes below hundreds of thousands of chips, FPGAs are more economical than ASICs. FPGAs also offer the possibility to fix bugs in the field by updating the FPGA programming.

Leaders in the FPGA field are Xilinx (www.xilinx.com) and Altera (www.altera.com). Since FPGAs (like all hardware logic) are best at parallel processing tasks with fixed functionality, some products combine regular processor cores with FPGA fabrics. The processor core takes care of unpredictable processing, while the FPGA part is used to accelerate processing suited to hardware implementation.

V. COMPUTER ARCHITECTURAL CHARACTERISTICS

Embedded computer architecture is much more diverse than general purpose desktop architectures. While it is to some extent true that technology trickles down from the PC/server market to embedded, there are many unique innovations occurring in the embedded market.

A. Instruction Sets

One peculiarity of the embedded market is that old architectures never die. Even in 2004, there will be billions of 8051s, Z80, and PIC chips sold: all 8-bit machines with instruction sets dating back to the 1970s. Also, the embedded market has given a second life to RISC architectures like MIPS that have faded from the general-purpose computer market. Furthermore, instruction set design goals and trade-offs are somewhat particular.

Code size is an important factor in most embedded designs, and instruction sets are designed and extended with code size in mind. Fairly typically, the NEC V850 (www.nec.com) architecture uses 16-, 32-, 48-bit, and 64-bit instructions to encode a RISC-style instruction set. The 32-bit ARM and MIPS architecture have been extended with reduced 16-bit instruction sets in order to reduce the code size. Instructions that perform a lot of work, like loading multiple values from the stack, are popular to reduce code size.

Instruction sets are also extended with instructions to accelerate particular processing tasks. For example, Motorola's (www.motorola.com) 68300 processors have a special table lookup and interpolate instruction to accelerate engine control tasks.

Of particular note are the *DSP (Digital Signal Processing)* instruction sets. DSP processors are specialized to performing data-plane processing, designed to take advantage of the regular structure of most interesting program loops, and with special instructions for common tasks (for example, FIR filters can be implemented with a single instruction on a TI C65).

The most extreme example of application-adapted instruction sets are offered by the *configurable processors*. Tensilica and Arc are the leaders in this field, where processor cores are customized by selecting which particular instructions to include in a particular configuration. It is also possible for the user to create new instructions to accelerate particular tasks.

B. Memory Systems

Embedded systems often feature quite complex memory systems. Caches are normally quite simple, featuring single-level instruction and data caches. However, the caches are often complemented with *tightly-coupled memories (TCM)*, fast on-chip memories that are under programmer control and not automatically managed by the cache system. Compared to caches, TCMs are more predictable (good for real-time systems) and use less power (thanks to reduced complexity). For IP cores, the size of caches and TCMs is usually configurable.

Memory is preferably kept on-chip to reduce power consumption and product cost: adding external memory chips is fairly expensive both in terms of production cost and power consumption. Usually, code is kept in ROM or FLASH memory on-chip, with a much smaller RAM memory for storage of variables and stacks (most embedded systems follow the classic "Harvard" design of separating code and data physically). EEPROM or FLASH memory is used to keep persistent data when the system is powered off. High-end systems require off-chip memories: even today, it is hard to fit

more than a few megabytes of memory on-chip.

On 8-bit and 16-bit architectures, pointers are limited in size. To accommodate larger memories, there is usually a hierarchy of memory areas and pointer types. There might be a *zero-page* memory which is addressed using an 8-bit pointer, with a "near" memory with 16-bit pointers, and various forms of "far" memory with 24- or 32-bit pointers. This leads to a programming model where the programmer needs to be aware of the allocation of variables to memory in order to efficient programs.

C. Pipelines

Pipelines in embedded processors tend to be simpler than their desktop counterparts, since the goal is to provide adequate performance with minimal cost and power consumption. For example, the ARM926 core requires up to 0.9 mW/Mhz; while the Pentium 4 uses about 35 mW/Mhz. The cost to go from acceptable to maximum performance can be very high!

8-bit and 16-bit processors are usually not pipelined at all, while 32-bit processors use everything from the simple 3-stage pipeline of the ARM7 to complex multiple-issue out-of-order pipelines on 64-bit MIPS systems. Currently, most 32-bit machines have moderately complex pipelines with 5 to 7 stages and strict in-order issue. Every generation of embedded processor cores tend to add a few more features in order to add processing power, but it is always done within the constraints posed by cost, size, and power consumption.

Since the tasks requiring the most processing power are usually well-known, they can be more efficiently solved by using hardware accelerators or special processors instead of a faster general-purpose processor. This leads to the classic RISC-DSP split of processing in mobile phones, and the use of special acceleration hardware for tasks like MPEG decoding.

There are also some truly extreme designs in the embedded field. Xelerated's (www.xelerated.com) network processors use a pipeline more than 1000 stages deep to efficiently route and filter network packets. Texas Instruments are successfully selling eight-wide VLIW DSP processors for customers requiring very high performance signal processing.

VI. SUMMARY

This talk has given a quick overview of the embedded computer architecture field. I have tried to give a feeling for the broadness of the embedded systems field and the wide range of particular computer designs that have been created to correspond to the many peculiar needs of the various end-user applications.

The main fact to remember is this: there is no typical embedded system, and any computer architecture feature ever invented is bound to have a valid application somewhere in the embedded systems field.

REFERENCES

- [1] Jim Turley, "The Two Percent Solution", *Embedded.com*, Dec 18, 2002. <http://www.embedded.com/story/OEG20021217S0039>