Chapter 2
Embedded DSP Devices

Serendra Reddy

Abstract As a consequence of the rapid surge in digital signal processing (DSP) technologies, DSP components and their specific algorithms continue to find uses in broad application areas, including the embedded systems arena. Embedded systems generally refer to systems that include dedicated hardware and computationally specific software. When several fundamental components of an embedded system are integrated onto a single silicon substrate it is referred to as a system-on-chip (SoC). These embedded systems, including SoCs, can either stand-alone or seen as a subsystem of a much larger and/or complex system. However, these systems are not without constraints, and constantly need to adapt to the drawbacks associated with limited hardware, restricted computational power and fewer resources. Recently, there has also been an increased interest in the use of field-programmable gate arrays (FPGAs) and application-specific instruction-set processors (ASIPs) within embedded DSP devices. This can be seen as a trade-off between size, speed and flexibility, with the latter being the driving force. Embedded DSP devices have proliferated through society so much so that we have become virtually oblivious to their impact. Among the countless applications of embedded systems, some products that require a DSP component include our mobile phones, digital radios, digital televisions, digital satellite set-top boxes, DVD players, MP3 players, heart-rate monitors, GPS navigation devices and automotive control systems. This chapter gives a brief introduction into the theory of DSP, followed by a more detailed examination of the architectures, implementations, security and applications within real-time embedded systems.

S. Reddy (✉)
Department of Electronic Engineering, Durban University of Technology, KwaZulu-Natal, South Africa
e-mail: serenr@gmail.com

K. Markantonakis and K. Mayes (eds.), *Secure Smart Embedded Devices, Platforms and Applications*, DOI: 10.1007/978-1-4614-7915-4_2,

2.1 Overview

A signal can be described as information within a form of detectable energy that is generated by a physical occurrence, like changes in electromagnetic radiation or air pressure. In order to investigate these signals this energy is first converted into a continuous electrical signal using, for example, a photosensor or microphone, as in the case of light or sound, respectively. These continuous electrical signals are commonly termed analog signals and the variations in these signals are represented by voltage values that are theoretically infinite, both in amplitude and precision. A digital signal is then the discretisation of an analog signal i.e. the representation of the continuous signal by a discrete (non-continuous) set of quantised values. This is achieved by taking samples (measuring the amplitude/voltage) of this continuous signal at successive non-zero time intervals i.e. a snapshot of the relative space-time. This process of conversion from an analog signal to a digital signal is called an analog-to-digital (A/D) conversion.

Digital signals can be represented in multiple dimensions, like one-dimensional, in the case of sound, and two-dimensional (2D), in the case of images. Although photons hitting a photosensor (like a CCD array in a digital camera) arrive at the speed of light, the image (or photograph) is merely a representation of the individual voltage levels on all the sensors at a single instance in time, arranged and stored in a 2D matrix. Digital video can then be extrapolated as a collection of 2D matrices captured sequentially, like at 0.033 s intervals in the case of a standard 30 frames per second movie, hence the term motion- or moving-picture.

If not clearly defined, the acronym DSP is often ambiguous, as it can describe the specific hardware/software processes used in handling digital signals, as well as a hardware processor. In this chapter, DSP refers to digital signal processing (DSP), which is the generic term applied to the hardware/software processing of digital signals and data by digital electronic devices. Digital electronic systems can range from super-computers, desktop computers and laptops, to tablet computers and smartphones, to small DSP specific systems and SoCs, including application-specific integrated circuits (ASICs), application-specific instruction set processors (ASIPs), field-programmable gate arrays (FPGAs), general-purpose DSP processors (GP DSPs) and general-purpose microprocessors (GPPs). The aim of the processing is to analyse the information content of the cached or stored signal data and sometimes modifying the signal depending on the desired output. This can range from, among others, noise reduction to data enhancement to data compression to pattern recognition, and to whether the system should operate in real-time. Real-time systems can be defined as those systems that respond in a timely manner to external actions or triggers [26]. In DSP, this implies that an output is produced from the current set of data before the next set of input data is collected and/or available to process. Once the digital signal has been processed and possibly enhanced (or altered) by the DSP system, it might be required to be sent back out into the real world, as in the case of music being outputted from a digital amplifier or equaliser. This process is, fundamentally, the reverse of an A/D conversion, where discrete digital data is

transformed or converted into a continuous analog signal, and is, therefore, called a digital-to-analog (D/A) conversion.

An embedded system is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a dedicated function [13]. Unlike a personal computer which serves a general-purpose, an embedded system is generally designed to serve a specific purpose, and is usually limited in size, cost, power consumption, processor speed, memory and hardware functionality. An embedded system can also be seen as being a part of larger system, containing possibly a collection of smaller embedded subsystems, including those with DSP functionality, each responsible for a separate task, like handling decompression and decoding of audio files on a MP3 player or the capture, compression and memory card storage of images on a digital camera. An embedded processor is a specialised processor, like a GP DSP or ASIC, designed to meet the requirements of a specific application, i.e. the functionality is often limited or tailored to just the intended purpose, e.g. with perhaps low power consumption and low heat dissipation, and restricted clock speed [26]. In order to handle real-world analog signals, including speech, images, video and music, an embedded processor must interface with external hardware such as input/output (I/O) devices like coders/decoders, memories and displays.

An embedded DSP device is typically a combination of one or more individual pieces of hardware (or subsystems) integrated into a single stand-alone system performing a specialised function, that requires a DSP specific hardware processor, like an ASIP and/or a FPGA, that uses DSP specific software algorithms and techniques to process and/or transform real-time input signals into a desired output.

2.2 Digital Signal Processing

Almost all information in the physical world is represented in the form of an analog signal, and as a result the processing of these analog signals represents a fundamental component within the field of electrical engineering. This subfield came to be known as analog signal processing (ASP); and entails the use of analog hardware in the manipulation of these signals. However, ASP had its shortcomings in the form of complicated electronic circuitry, inflexibility, varying accuracy and inconsistent reproducibility. In addition, sophisticated applications, like speech and image processing, were not suitable to ASP technologies. These challenges needed to be addressed and were eventually solved by the advent of digital systems and DSP.

Although the mathematics of DSP algorithms had been in existence for many years, it was only the emergence of the GPP and GP DSP in the 1970s that marked the turning point in digital systems. The original systems were primarily fixed-point machines [39, 45]. The mid- to late 1970s saw the introduction of floating-point machines and together with supporting memory devices gave rise to the era of DSP, which by 1980s included multi-processor systems with massively parallel

architectures, allowing for efficient execution of the fast Fourier transform (FFT)[1] and vector-based processing schemes. However, non-conventional schemes, like adaptive and high-resolution signal processing remained a bottleneck [18, 49], until recently. Advances in super-scalar and massively parallel processor technologies have seen the GP DSP go from being able to perform several hundred million multiply accumulates (MACs) per second (or about 21 ns per MAC) in 2000 to around 5 billion MACs per second (or about 3 ns per MAC) in recent years [22, 27].

2.2.1 The DSP Processor

The core purpose of a DSP processor (or GP DSP) is to perform signal processing, and almost every single DSP application is based on efficient mathematical implementations of one or more of algorithms [15, 36, 37] shown in the following table:

Fourier transforms are used for representing signals in the frequency domain; convolution can be used to perform filtering in the spatial domain; correlation is used to detect similarities in signals, like in the case of Radar; finite impulse response (FIR) and infinite impulse response (IIR) filters can be used in noise attenuation and other frequency selective processes; 2D Fourier transforms are used for image processing in the frequency domain, and discrete cosine transforms are used in image compression, like JPEG.

The DSP processor was thus optimally designed around the ability to efficiently execute the above algorithms. This was done by exploiting the inherent similarities between the algorithms, like the summation and multiplication operations. The combination of the summation, which can be described as a "for" loop in software, together with the multiplication operations, results in the accumulation of a large number of multiplied elements. It was, therefore, logical to develop a processor that was able to resourcefully accommodate the operands of multiplication and accumulation. In addition, there are intrinsic structures in these algorithms which allow for non-dependent parts to be operated on separately. In the end, the common factors observed in the digital signal algorithms allowed for the tailoring of a DSP specific processor that could achieve tremendous execution savings.

The GP DSP differs mainly from the GPP in its memory architecture, internal architecture and instruction set. The memory architecture of the GPP is based on the Von Neumann single memory model, whereas the GP DSP is based on the Harvard dual memory model (Refer to Fig. 2.1), which is designed for parallel access to the program and data memory, allowing for the fetching of multiple data and/or instructions at the same time. An advancement to the Harvard architecture is referred to as the super Harvard architecture and includes an instruction cache and dedicated I/O controller (with DMA) [42]. The internal architecture contains several multipliers and accumulators, in order to optimally perform fixed-point and floating-point

[1] The discrete Fourier transform (DFT) is the method of translating any sequence of discrete values into its frequency domain equivalent, by representing the signal as a composition of sine and cosine waves. The FFT is the more efficient method of generating a DFT.

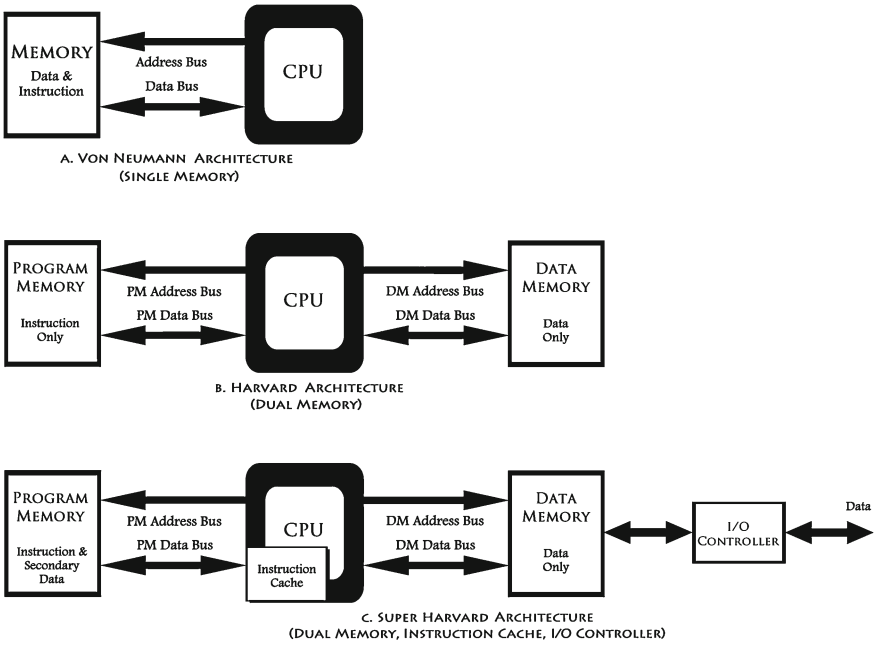

Fig. 2.1 Microprocessor architecture

multiply-accumulate (MAC) operations extremely fast, which is a necessary requirement in order to efficiently perform the DSP algorithms described in Table 2.1. The specialised instruction set was designed to maximise the use of hardware, minimise memory space and increase efficiency. In addition, it incorporated measures to alleviate some of the problems associated with the limitations of working in the digital arena, like quantisation errors, round-off errors, finite wordlength effects and overflow. All this, in the end, means that a GP DSP performs signal processing much more proficiently than a GPP, which is optimised for non-signal processing centric applications.

Table 2.1 Common DSP algorithms

Discrete fourier transform (DFT)	$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}$
Convolution	$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$
Correlation	$r_{xy}(n) = \sum_{k=0}^{M-1} x(k)y(n-k)$
Finite impulse response (FIR) filter	$y(n) = \sum_{k=0}^{M} b_k x(n-k)$
Infinite impulse response (IIR) filter	$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$
2-D DFT	$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$
2-D Discrete cosine transform (DCT)	$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)cos(\frac{\pi(2x+1)u}{2M})cos(\frac{\pi(2y+1)v}{2N})$

2.2.2 The Real-Time DSP System

A non-real-time system [23, 24] technically does not have any limitations on the amount of processing or execution time required to complete a task. A real-time system can be seen as having an environment in which the correctness of the system depends not only on the logical results of the computation, but also on the time in which the results are produced [43].

The definition of real-time systems with respect to DSP are not entirely unambiguous, hence in this chapter, a real-time DSP system [23, 24, 27] refers to a system that generates an output signal within the rate of which the input signals arrive, i.e. the ability to process one sample within the duration it takes two consecutive samples to arrive. Real-time DSP places rigorous demands on both the hardware and software design of the system so as to achieve the predefined tasks within the allocated time frame. Some of the numerous applications that employ real-time DSP include Radar [5, 45], Sonar [45, 46], signal intelligence [48], speech [39, 41] and image [22, 38, 45] processing and missile guidance [48].

Figure 2.2 illustrates the basic functional blocks of a real-time DSP system, where a physical analog input signal is converted to a digital signal, which is then processed by a DSP processor, and eventually converted back to an analog output signal. The following is a brief discussion of the function of each stage in the system:

- Input signal. An electronic sensor (microphone, photosensor, etc.) will first convert the variances of the analog signal (temperature, pressure, sound, light, etc.) into an electrical signal.
- Analog signal pre-processing. The aim of this circuitry is to perform some pre-processing on the incoming electrical signal. This can include amplification, voltage regulation, anti-aliasing filtering and possibly other analog enhancements that could benefit the A/D conversion.

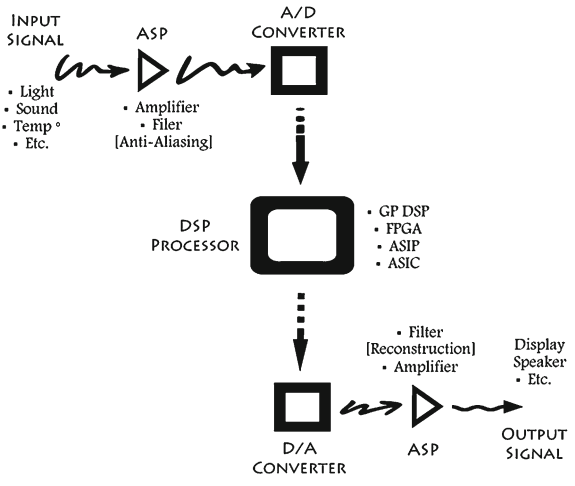

Fig. 2.2 Functional blocks of a real-time DSP system

- Analog-to-digital convertor (ADC). The proverb, "a chain is only as strong as its weakest link", when applied to a real-time system, is applicable to the ADC, owing to the fact that the sampling rate defines the digital filter's working frequency. There are three components that comprise the ADC: sample-and-hold (S/H), quantisation and coding. In order to obtain the digital representation of the electrical amplitude, the electrical signal is first sampled, which is the process of tracking and tapping into the fluctuating electrical signal; this tapped value is then held constant for a single A/D conversion cycle, in order to be read and translated (or digitised). N.B. For a complete and unambiguous reconstitution of a continuous analog signal by a digital process the electrical signal must be sampled at a frequency that is at least greater than twice the highest frequency of the respective signal. This has been defined by the sampling theorem [37]. Quantisation is the process of mapping an electrical value of theoretically infinite precision to a value of finite precision. This is dependent on the quantisation step size, which is the minimum significant value allowed i.e. the value to which the discrete time signal must be rounded up or down to. A system that has, for example, only one significant digit after the decimal point will result in the values 0.4862969 and 2.348672 to be rounded to 0.5 and 2.3, respectively, with a quantisation error of 0.0137031 and −0.048672, respectively. The final phase is the coding of the quantized values into binary, which is, like the quantisation stage, also dependent on the significance of the system. In order to maximise efficiency, the resolution, or step-size, of both the quantiser and encoder are jointly optimised. The errors due to quantization and wordlength effects can be modelled to determine the effective accuracy and throughput of the ADC.
- DSP processor. This represents the hardware and/or software responsible for the analysis and/or modification of the digital bits or signals by DSP methods; a process which largely involves the application of at least one of the algorithms given in Table 2.1 (Sect. 2.2.1).
- Digital-to-analog convertor (DAC). The transformation from the digital world to the real world is done at this stage. A voltage is generated at the output, proportional to an electrical signal, corresponding to a binary word input to the DAC. In order to go from a series of discrete values to a continuous signal requires a type of interpolation. Although there are several methods of interpolation [37], most DACs are zero-order-hold. In this system a constant voltage value is outputted until another sample value is received; the result is a staircase effect. In order to create a continuous smooth analog output signal, the output from the DAC is processed through a post-filter.
- Analog signal post-processing. This represents the smoothing and anti-imaging filter. The final reconstruction of the analog signal is done by smoothing off the corners of the staircase signal generated by the interpolation or zero-order-hold process at the output of the DAC. In addition, the interpolation could cause image (high frequency) components being passed above the folding frequency, resulting in replications (or images) at multiples of the sampling frequency (sometimes referred to as post-aliasing [37]); this could generate possible degradation of the output signal due to overloading of certain components of the external circuitry, and thus are subsequently filtered and removed at this stage.

- Output signal. The final converted, modified and/or improved analog signal is processed at this stage, usually by some human-compatible real-world device, like a speaker, display or computer.

Real-time systems are occasionally influenced by response time constraints, like unanticipated delays or latencies. The effect of these inconsistencies on the overall system is vitally important, consequently creating two types of real-time systems viz. soft and hard [33]. Soft type implies that in the event of a missed deadline there is a degradation of performance, but no system failure; hard type subsequently implies a hard deadline, which if not met, results in system failure e.g. the navigation system of an aircraft.

Real-world embedded DSP systems are usually qualified as hard real-time systems. Compared to desktop programming, embedded DSP real-time systems must be able to respond predictably and reliably to all external events, meaning that the DSP code must not only execute as predicted, but execute on time, without little or no exception.

2.2.3 The FPGA in DSP

In the past, whenever there was a need to meet extreme performance requirements, and a programmable device could not handle the load demand, or there was a solution requiring ultra-low power ultra-small silicon size, an ASIC was the only alternative for an embedded DSP system [27]. ASICs are produced by directly mapping algorithms to an integrated circuit; this, however, provides extremely limited flexibility, and virtually no room for changes once fabricated. In addition, as DSP applications become vastly more complicated, the mapping of the DSP functions to circuits become vastly more difficult.

As a result, over the past decade, the DSP market has seen a drastic increase in the use of alternate processing systems, most especially the FPGA [12]. The FPGA is seen as a compromise between the rigid ASIC and the programmable GP DSP. Although both the ASIC and GP DSP have their own merits, the attractiveness of the FPGA comes in its flexibility, or configurable hardware, which has shown to have a cost/performance of between 10 and 25 times that of a typical high performance GP DSP [11, 47].

Although GP DSPs have conquered the world over, the requirements of certain newer applications tended to exceed their processing capabilities. The FPGA was able to meet these demands, owing to its ability to reconfigure the logic, thus allowing for the design of computationally effective structures, in the form of special purpose functional units that can perform limited tasks very efficiently [34]. The massive computational resources, as well as the ability to configure highly parallel architectures, surpassed the throughput of even the high-end GP DSPs. Due to the nature of FPGAs, field upgrades of the configuration file are rare; whereas, GP DSP code and product software patches and updates are widespread. Since a FPGA

is hardware-programmed, by manipulating logic at the gate level, it is possible to construct DSP-oriented processors in parallel that can efficiently and simultaneously solve a desired DSP task.

The downside to the FPGA is the effort of complexity required to map DSP applications, including ADCs (although significantly higher frequency ADCs can be achieved with FPGAs), DMA controllers, bus interfaces, etc.; requiring architecture implementations at the register-transfer-level (RTL)[2] usually via some hardware description language (HDL)[3] like very-high-speed integrated circuits HDL (VHDL). This complication is, however, being addressed with steadfast advancements in FPGA tools (including simulators) and libraries. Another issue concerning the FPGA, is working with floating-point cores, which can consume a significant amount of area within the FPGA, especially when parallel processing is considered; one alternative is to settle for using integer-based coefficients, or to possibly pipeline the floating-point operations, which will reduce the required gate area, but at the expense of the response times. A final issue that is worth noting is the issue of "excess baggage"; the FPGA essentially implements a function by turning on or off different logic gates; the problem being that once an application is programmed into the FPGA there can be a lot of redundant gates taking up unnecessary space. These aforementioned issues are relative to each project and need to be considered in determining the tradeoff between performance and size, cost and development times.

2.2.4 The ASIP in DSP

Another avenue of interest that has recently emerged in the embedded systems sector is the use of embedded soft-cores. Embedded soft-cores can be seen as ASICs with application-specific parameterisable components, and are thus referred to as application-specific instruction-set processors (ASIPs) [32]. Although the instruction set of an ASIP is designed around specific application requirements that tailor the processor for these applications, it is also a programmable machine with a degree of flexibility, allowing it to run various software programs [27]. The former allows for a product that can be high performing while low in power consumption, and high volume manufacturing can be used to lower costs. ASIPs can even come with their own development environment, debug tools, simulators and compilers, with the option of adding peripherals for communications, I/O, memory control, etc.

FPGA-oriented-ASIPs [25] are also possible, whereby a "soft-core" processor can be configured and downloaded onto a FPGA and used just like any other embedded processor, including a GP DSP. A DSP ASIP mapped on an FPGA will consist of

[2] The RTL is the level above the transistor or logic gate level that translates the circuits described by the HDL into their equivalent sequential (usually consisting of registers comprising a number of D-type flip-flops) and combinational logic structures.

[3] A HDL, which is implemented at a level above the RTL, is a method of using text-based expressions to represent an algorithm that describes the behaviour of a digital circuit.

an instruction set optimised for a class of DSP applications; the reduced instruction-set-like architecture can take a form of a parallel process-unit array that can realise high processing speeds and high levels of parallelism [51], thus combining the programming capabilities of a GP DSP and the high throughput of an ASIC. Any type of microprocessor can be implemented on an FPGA; however, in recent times the vendors have provided soft-core processors specifically designed for FPGAs. These soft-cores include instruction sets, arithmetic-logic units, register files, and other features specifically optimised for the FPGA resources, and dually serve as preventative measures against inefficient and/or incorrect use of FPGA resources. It has been shown that the performance overhead between general circuit implementations on FPGAs versus ASICs is far superior when comparing overheads between such soft-core processor implementation on FPGAs versus ASICs [40]. A key value of an ASIP on a FPGA is that they are composed of smaller building blocks that can be reconfigured on the fly to implement more than one high-level function [25]. Example relevant to DSPs would be FIR filters and Fast Fourier Transform (FFT) blocks. Since these two high-level algorithms share many common sub-blocks, the ASIP can be easily altered to implement the FIR, instead of the FFT, in hardware, by changing the interconnect between these subblocks.

The customisable ability of the ASIP is not without challenges, as currently, each application that is compiled must be simulated and run on all possible configurations of the ASIP, which can be exponentially exhausting owing to the increase in configurations with the number of parameter values [17]. This can result in an increase in the time-to-market and subsequent cost, prompting a possible decrease in the number of customisable parameters or opting for a general-purpose chip solution. However, there are methods to improve these drawbacks, like having customisable development, simulation and test-bench environments, which aid in optimising the values of these architectural parameters. Although soft-core processing is still in its relative infancy, their applications on SoC platforms make them hugely attractive within the embedded systems market [17], thus creating a consistent drive to adapt and advance this technology.

2.3 Embedded DSP Systems

Both a general-purpose computer and an embedded computer (or system) can simply be seen as devices created to store, retrieve and process data; with the fundamental distinction lying in their application, size, power, cost and predictable speed. A general-purpose computer is typically a multi-tasking system that can respond to the requirements of multiple applications, such as playing music or accessing web pages or generating spreadsheets. An embedded computer might only perform one specific task, like the removing of noise from an audio stream. In addition, unlike desktop computers, embedded devices do not necessarily have a user interface; alternatively they might have a basic user interface, like a simple button with a LED, to a midrange user interface, such as an alpha-numeric liquid crystal display (LCD)

menu system, all the way up to a complex graphical user interface (GUI), like a touchscreen icon-based menu system on a mobile phone.

With the rapid evolution and requirements of modern dedicated embedded circuitry, the modern embedded computer is seen as a viable replacement to application-specific electronics, with the fundamental advantage being that the former can be used to define functionality using software and/or hardware, as opposed to the inflexible dedicated hardware of the ASIC [6]. The embedded system can subsequently be partitioned into the hardware component, which is responsible satisfying the performance requirements of the application, and the software component, which provides for bulk of the functionality and flexibility within the system [33].

In the case where there is a need for the handling of time critical tasks by an embedded system, a DSP component may be required. Given the high demands of current embedded devices, like smartphones, it has become an unavoidable necessity for most modern embedded systems to include a DSP subsystem.

A DSP subsystem can also be fabricated within an embedded GPP system together with additional components, like a direct memory access (DMA) controller, a programmable interrupt controller (PIC), a programmable general-purpose timer and even Ethernet interfaces. These self-contained platforms are referred to as system-on-chip (SoC) processors, and due to the reduction in the need for external peripheral devices the overall size and cost of a product can be optimised.

Embedded systems that incorporate DSP technologies include three-dimensional high-definition televisions (3D HDTVs), digital cameras, media players, digital voice recorders, fingerprint scanners, unmanned aerial vehicles (UAVs), software defined radio (SDR) and innumerable others.

2.3.1 The Embedded DSP Architecture

The architecture of an embedded DSP system is modelled on the architecture of a generic embedded system. Both comprise four basic units, including processor, memory, general-purpose I/O and bus subsystems, with the embedded DSP system containing a DSP processor and a few other I/O peripherals.

Figure 2.3 illustrates the basic architecture of an embedded DSP system. The following is a brief discussion of each subsystem:

- Embedded processor unit (EPU). This is essentially the brains of the operation, and its key responsibility is in the processing of instructions and data. The EPU can include a master processor (or several master processors) and can either have one or several associated slave processors (or none at all). A master processor can be classed as a stand-alone microprocessor or a microcontroller, with the latter representing a subsystem where one or more microprocessors are integrated together with other memory and I/O components [30]. Furthermore, the embedded DSP processor can be an add-on processor which accompanies a GPP within the processor subsystem, or a stand-alone processor representing the entire EPU. Depending on the application, the co-processor scheme can be advantageous, in the sense

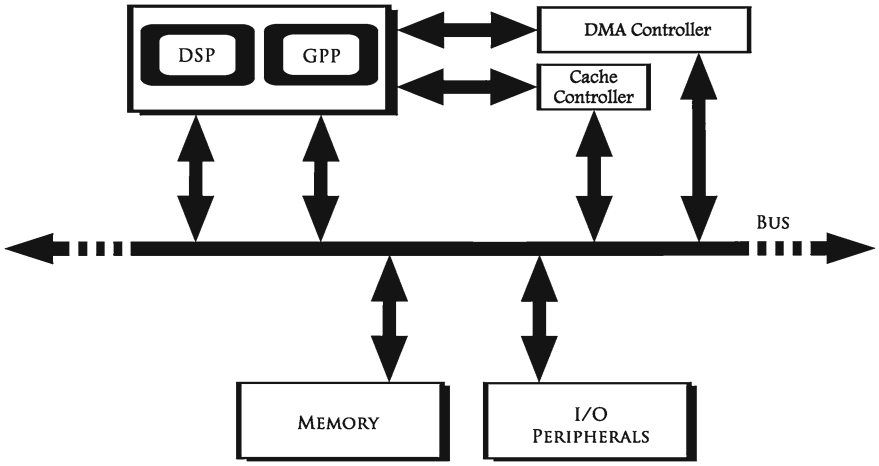

Fig. 2.3 Embedded DSP architecture

that the GPP can handle the miscellaneous and/or mundane tasks, as for example, hardware management and connection protocols, or interacting with a keypad or display, while the DSP processor focuses on the computationally intensive real-time demands. In addition, cache and DMA controllers (discussed later) can also be included as part of the EPU. These are, in essence, slave processors that function primarily to improve the overall performance and efficiency of the processor cores, and, therefore, when included, define the EPU as a microcontroller.

- Memory. This subsystem supports data and program storage for the EPU, and represents multiple levels of memories that range hierarchically from the local and/or cache memories within the processor cores, to on-chip memory (as in the case of SoC), to the extrinsic or off-chip main memory. These three memory architectures are called Level 1 (L1), Level 2 (L2) and Level 3 (L3) memories respectively, and vary incrementally in size, and decrementally in performance [21]. L1 memory is literally closest to the core on the silicon die and is fabricated for maximising interoperable speeds. Within the GP DSP this L1 memory is configured using the Harvard architecture scheme, where the instruction and data segments are split.
- Cache controller. Cache is essentially a small, fast advanced memory that holds duplicates of some of the data and/or instructions stored in main memory. The cache controller takes on a mediator role between the embedded processor and main memory. It works by simultaneously sending memory requests to both the cache and main memory. If the location requested is in the cache, it aborts the main memory request, while forwarding the location's content from the cache to the embedded processor [50]. Cache is a valuable L1 memory which is costly in terms of silicon size, and thus is limited in its capacity. The main aim of the cache is to maximise the instances of successfully finding what it needs in the cache as an alternative to having to wait to retrieve it from the larger and slower main memory [21]. DSP processors typically contain smaller and simpler caches than GPPs.

It has been shown that once the cache reaches a certain size there is a saturation of the performance benefits. This typically occurs between 256 KB and 512 KB [33]. DSP processors frequently use caches to free memory bandwidth rather than to minimise memory delays.

- Input/Output Peripherals. These interfaces are in charge of interacting with functional units that are connected to the outside world, with the primary responsibility of bringing data into, and getting data out of, the embedded system. They are principally managed by a slave processor known as the I/O controller which interfaces with the communication interface and I/O buses in order to facilitate data communication between the EPU and I/O device [29]. There are multiple roles performed by these I/O devices, which can be extremely diverse in their range, and consist of simple circuit components, like LEDs, up to other complex embedded systems. These I/O peripherals can be broadly categorised as follows.
  - Human-machine I/O: Keypad/keyboard, touchscreen, mouse, etc.
  - Graphical and audio I/O: Camera, microphone, speaker, displays, etc.
  - Real-time I/O: ADCs and DACs.
  - Communication and networking I/O: IEEE 802.11 g/n, Ethernet, etc.
  - Data transfer I/O: USB 2.0, IEEE 1394 (Firewire), etc.
  - Subsystem controls: Timers, counters, low-speed serial interface, etc.
  - Storage I/O: Optical disk, Flash, SDRAM/DDR, etc.
  - Debugging I/O: JTAG, parallel and serial ports, etc.
- Bus. The bus is the mechanism responsible for the interconnection and exchange of data, address and control signals between all subsystems. It does not just physically represent a collection of related wires connected between functional units, but also conceptually represents a set of protocols that are used to allow for communication between the EPU and memory, EPU and I/O, and memory and I/O. The embedded bus network can be split into three types of busses via system, backplane and I/O buses [29]. The system bus interconnects the EPU with the cache and the main external memory. The backplane bus represents a single bus that connects all three subsystems. The I/O bus is an extension of the system bus and handles communication between the I/O peripherals and the system bus, including interrupt requests. The performance of a bus is measured in bandwidth, which is dependent on the design, protocols, number of bus lines and the interconnect permutations of the respective bus.
- DMA controller. DMA is a bus operation that provides for the movement of data between subsystems without the need to interfere with the more important tasks being performed by the processor, thereby improving overall system performance. The DMA controller essentially oversees the DMA operation. This is done by first requesting control of the bus from the main processor followed by the transfer of data to and from IO and memory, or ports and memory, or internal and external memories [33]. By off-loading memory transfers from the processor it allows for efficient parallel processing. Without DMA the main processor is consumed for the entire time required for the bus transfer, and since the buses often function at a reduced clock speed compared to the main processor, this can significantly reduce

overall efficiency of the embedded system [50]. In a DSP, for example, when data comes onto the serial port from the A/D converter, rather than interrupting the DSP processor to transfer the data to memory (and subsequently consume a large amount of processing cycles), this can be handled by the DMA controller; the DSP processor can then focus on the execution of the algorithms while the DMA controller is handling the movement of data [33]. This feature of overlapping operations, via DMA, is extremely common in real-time embedded DSP systems.

2.3.2 The Embedded DSP Processor and RISC

At the lowest depth of any processor is ultimately a collection (albeit millions) of transistor-based hardware gates. These gates are connected into groups of combinational and sequential logic circuits that perform the instructions of a higher level application. These logic circuits are in the end the essence of the machine, and operate using binary language. This binary or native language is known as machine language. The problem encountered is that application layer instructions are formulated at a non-binary level, and therefore a process of translation between this level and the lowest level hardware is required. This hierarchical process contains several intermediary stages with the most crucial sublevel being the Instruction set architecture (ISA).

The functionality of any application is determined by a collection of interdependent instructions. These instructions can be constructed by a myriad of high-level languages like C or FORTRAN. ISA is the common platform that provides for the interpretation and execution of high-level instructions independent of the high level language employed [10]. This is achieved by first having all high-level language instructions compiled into a single universal language. This universal language is known as assembly language and uses mnemonics to express instructions. These assembly language mnemonics are then finally encoded into the binary machine language.

At the ISA level there exists two design categories complex instruction set computing (CISC) and reduced instruction set computing (RISC). CISC as the name suggests is the system of employing complex or complicated instructions, like those used when working directly with multiple array elements. RISC, on the other hand, is the system of employing reduced or simple instructions, as in the case of just adding two integers.

An early downside of complex instructions was that there was a need for complex and expensive complementary hardware to carry out these instructions. To work around this problem, a micro-programmed computer was introduced [10]. This came in the form of a small run-time interpreter, which was located between the ISA level and the hardware, which converted complex instructions into simple ones. This, therefore, eliminated the need for complex hardware, and meant that complex instructions could be executed on simple hardware.

An inherent shortcoming to RISC systems was that they were first constrained to a set of simple instructions, and second these limited operands were restricted to the processor's internal registry, as opposed to the main memory. CISC, on the other hand, does not have such restrictions and allows for an increased volume of complex instructions, which can be placed in the main memory.

With the staggering increase in functionality and features demanded of our modern embedded DSP systems, comes the need for higher processing capabilities, increased number of chips per system, and associated power constraints. Within the embedded systems environment, each of these issues comes with a whole host of related problems, ranging from processor size, to access to a compatible, sufficient and sustainable power supply, and heat dissipation. Heat dissipation is not only an aesthetic concern, but also of vital importance to both processor performance and lifespan. In addition to these issues, a problem that has been encountered is in the advancement of battery technologies, which have not kept abreast with the increase in consumer expectations for more functional and feature driven smart embedded devices.

As a consequence of the aforementioned problems, the hardware for these specialised high-demanding embedded DSP systems had to adapt almost independently; as a result a separate evolutionary branch emerged in conjunction with their generic computer counterparts, and prompted the rise of RISC.

Although CISC has some inherent advantages, the complexity of the instruction set requires additional processing, hardware and memory, prompting an increase in physical size and power needs, which consequently is inadequate with regard to embedded systems. Desktop and server computing usually do not suffer so severely from power requirement issues and size constraints, and these systems have the advantage of large cooling units, including heat sinks and fans, and air-conditioned environments.

RISC systems, having instructions that are simple in nature, can be executed directly on the simple hardware, thus eliminating the need for any additional middleware, and as it turns out, the confinement of the register-based operands not only results in a simplified processor design, but additionally creates improved application performance. This, in the end, means that RISC is ideally suited for embedded systems.

Applications, like video encoding, that require high computational complexity as well as data bandwidth, can be designed using just a DSP core; however, higher performance would entail either an increase in the clock frequency or the use of multiple DSP processors. The disadvantage of these alternatives is increase in silicon size, which subsequently results in the rise in cost and power demand [28]. A better approach would be to have a SoC design that combines a DSP processor with a RISC processor [14]. This kind of design can split tasks thus providing video-specific hardware acceleration necessary for optimal encoding and decoding. The comparative trade-offs between possible SoC architectures that can be employed in video encoding are shown in Table 2.2 [28].

Currently, the most popular RISC processor architecture employed in embedded DSP systems is the Advanced RISC Machine (ARM) [4, 7]. SoC platforms for these high-performing embedded DSP devices can consist of multi-core ARM processors,

Table 2.2 SoC architecture

Solution	Programmability	Performance	Power	Cost (Area)	Development time (Reuse)
AISC	Low	High	Low	Low	High
FPGA	High	Medium	High	High	Medium
Multi-Processor + Multi-Core	High	High	High	High	Medium
DSP	High	Medium	Low to Medium	Medium	Low
DSP + Co-Processor	High	High	Medium	Low to Medium	Medium

together with multi-core DSP chips, and various other interfaces. These ARM/RISC processors feature a highly specialised and optimised architecture, with extremely low power consumption [3]. There is, currently, a whole subdivision dedicated to DSP processors using the ARM architecture. A possible configuration for the above video encoding problem could be to use an ARM 32-bit RISC processor which could be extended into an efficient co-processor scheme that offers a standard ARM processor integrated with a DSP-oriented data path and an associated DSP instruction set. This provides for a small low cost embedded DSP chip that is optimised for performance while providing low power consumption [9].

2.3.3 Embedded DSP and Security

The security issues that affect large computer systems also apply to the embedded systems environment. However, the latter comes with added complexity, due not only to the limited hardware and software capabilities, but also because of the diverse environments which these devices are deployed.

Security concerns are especially crucial in embedded DSP systems that are involved in safety critical hard real-time functions, like automotive breaking systems, and in protection of private and confidential information using encryption, as needed for cellular voice and data transmissions.

The malicious exploitation of sensitive data as well as catastrophic system failure (owing to internal or external influences) can signify severe security risks apparent in using an embedded device. Among the numerous possible security threats inherent in embedded systems [44] there are four types that can be considered crucial when considering an embedded DSP device:

- Physical/Environmental. Considered when there exists a potential vulnerability in the manipulation of the physical components of the device, thereby either causing

the device to malfunction or fail completely. This can also include complete device destruction.

- Internal. Involves the breakdown of the hardware and/or software of the embedded system, as in the case of a processor overheating or an erroneous/corrupt software routine, thereby causing erratic/incorrect operation and/or complete failure of the device. Owing to the often unpredictable nature of hardware malfunction it is common practice, especially in large-scale production, to perform some form of system/load testing prior to full-scale production. The more reputable embedded systems designers may also follow a form of best practices and software lifecycle management in order to improve the success of their development projects.
- External. Involves hacking, hijacking and/or purposely corrupting the embedded device, usually, but not always, by some form of software tampering. Off-the-shelf PC security products, like anti-virus and firewall, are designed to be general and often very flexible. However, application dependent embedded devices are unique and specific, and therefore a universal solution is implausible. A system cannot be guaranteed secure even when using cryptography, as employing the most cryptographically astute algorithm is almost rendered futile should someone be allowed to access the information, directly from the source, prior to encryption [19]. Another type of security violation can involve a denial-of-service attack, whereby, as in the case of a mobile phone, the signal can be "jammed" so as to prevent the transmission or reception of calls.
- Information/Data. Involves the access, interception and/or decryption of stored and/or transmitted information. As highlighted in the previous point, software-based encryption usually has some kind of security infirmity due to the high risk environment inherent when managing the certificates/keys; an alternative might be to employ the use of an embedded DSP device that incorporates an on-board encryption module which can store and encrypt sensitive information [19]. As in the case of cellular communications, the GSM speech service is relatively secure, up to the point of entry of the network provider; however, this can be construed as a potential security vulnerability, as the cellular network provider, and not the subscriber, is controlling the encryption/decryption. A malicious intrusion of the network provider's system could therefore render all subscribers susceptible to privacy violations. A possible solution will be for the subscriber to incorporate some form of embedded DSP device that can provide personalised encryption prior to the speech entering the handset, thereby providing exclusive protection [20]. Not that DSPs generally do not have the attack resistant capabilities found in security modules (such as smart card chips).

However, with the role of the embedded DSP device in many vulnerable applications, especially hard real-time safety-critical systems, it is vitally important that a thorough risk versus reward analysis be undertaken when pondering the repercussions of a security failure.

Table 2.3 The mobile phone evolution

Mobile phone: features/applications before 2000	Mobile phone: features/applications between 2000–2005	
	• Camera	• GPRS
	• MMS	• 360 KB RAM
• Voice	• Push-Email	• Colour touchscreen
• Keypad	• Monochrome touchscreen	• 128 MB Memory
• Basic monochrome display	• QWERTY keyboard	• Video camera
	• 64 MB memory	• Colour LCD display
• Short message service (SMS)	• FM radio	• Extended battery life
	• Infra-red	•Basic games
• Monochrome LCD display	• Calendar	• 3G
• WAP	• Video playback	• Size (Thinner)
• MP3 Playback	• Bluetooth	• HTML browsing
	• Size (Shorter)	• 1 Mega-pixel camera
	• Colour screen	• Windows mobile
Mobile phone: features/applications between 2006–2010		
• Dual-processor	• GPS navigation and google maps	• GPU
• High-resolution LCD screen	• Voice-over-IP	• High-resolution graphics gaming
• Multi-format document viewing	• Downloadable applications (Apps)	• High-definition video capture
• Multi-touch sensor screen	• Document/office editing suites	• 1.2 GHz ARM processor
• Instant messaging	• Programmable open-source O/S	• 1 GB RAM
• Instant messaging	• Resistive touch screen	• 3D autostereoscopic displays
• Accelerometer	• 8 Mega-pixel camera	• 3D stereoscopic capture
• 5 Mega-pixel camera	• Smile and blink detection	
• 8 GB memory		

2.3.4 Embedded DSP and the Mobile Phone

The worldwide ascension of the mobile phone has been met with significant evolutionary changes to the capabilities of the device; having transformed from a wireless gadget that made and received phone calls, into a handheld computer integrated with a mobile phone (known as a smartphone) containing myriads of features and seemingly limitless capabilities (Refer to Table 2.3).

In the original mobile phone the DSP processor was connected to the audio and RF interfaces and responsible primarily for modulation, demodulation, decoding, encoding, encryption, filtering and noise reduction. The GPP was responsible for hosting the operating system and controlling the RF modem, user interface/keypad and few other control functions. The roles of these processors had to adapt significantly to

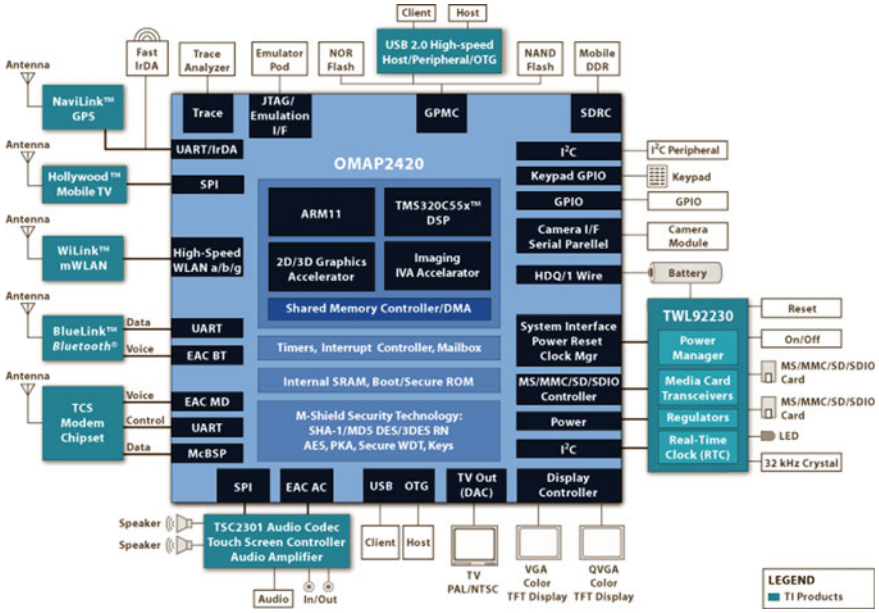

Fig. 2.4 Smartphone SoC processor (courtesy of Texas instruments)

meet the new demands of the modern mobile phone. As a consequence, the chip manufacturers had to find innovative ways to develop multi-integrated SoC processors for products that were smaller, lighter, more energy efficient and richer in features.

These processors, like the smartphone SoC processor shown in Fig. 2.4 [31], were designed for complete solutions in one embedded system. The embedded DSP system in Fig. 2.4 has a co-processor EPU that includes a RISC/ARM11 quad-core processor and a DSP multi-core processor. These process the basic telephony functions but also handle a multitude of additional peripherals, including GPS navigation, 3G and WiFi network connectivity, high resolution touch screens, and cameras, as well as support security. The EPU also interfaces with an image accelerator for image and video processing and video decoding, and a 3D graphics accelerator for gaming and other 3D applications.

In such an embedded system the DSP processors or processor cores have to perform several functions, a few of which include:

- Security: A DSP multi-core processor can be integrated in a SoC to perform the function of a security module that contains a cryptographic engine responsible for the logical protection of the voice, data and internetworking systems, as well as providing support for the defense against malicious software and firewall acceleration [1, 2, 48].
- GPS Navigation: In the GPS link there will be a DSP-based multi-channel satellite receiver that handles the signals received from the satellites. The DSP processor

will then obtain the location parameters by performing specific mathematical computations on the received data [35].

- Voice-over-IP (VoIP): VoIP is a function whereby a user can make telephone calls over the data/internet network as opposed to the GSM network. Under certain conditions this could result in the user paying very little or nothing for calls. In this scenario the DSP will perform the low bit rate coding and encoding of the voice signals.
- Image and Video Capture: During image and video capture, a DSP processor will be used to control the display prior to capture and then perform image processing on the data stored in the buffer memory after capture. In addition, the DSP processor can provide for features like face, smile and blink detection.

No other consumer electronics device in history has had such a popular global impact, both socially and culturally, as the mobile phone. Since emerging onto the scene, it rapidly spread from a just handful of countries to around 200 by 2010 [16].

It must also be emphasised that with the evolution of the smartphone, there become an ever increasing need for security, as more and more personal and sensitive information is being stored on, and transmitted between, these devices. Looking to the future, it is not hard to see that the smartphone and tablet PC may displace a wide range of existing computing and sensing devices.

2.4 Discussion

DSP is pivotal in systems that involve speech, vision, high-fidelity audio, modulation-demodulation, image compression and compositing, beamforming, echo cancellation, spectral estimation and real-time processing. These types of DSP subsystems have broad ranging applications within the embedded devices market and are incorporated extensively into both the military and commercial sectors.

With the demand for more complicated, secure, smaller, faster and cheaper real-time embedded systems, there will be a need to develop newer more advanced DSP techniques and technologies. However, it is also important to note that as the number of computationally expensive processes increase, so, too, does the power consumption. A collaborative effort between improved RISC processor designs and more energy efficient power sources is necessary in order for these systems to continue to evolve.

References

1. ADSP-2141 SafeNet™ DSP Security System on Chip. Analog Devices. http://www.analog.com/static/imported-files/product_highlights/ADSP2141_Brief.pdf (2000). Accessed 15 September 2011
2. ADSP-2141L Data Sheet. Analog Devices. http://www.analog.com/static/imported-files/data_sheets/ADSP-2141L.pdf (2000). Accessed 15 September 2011

3. ARM Architecture Reference Manual, 3rd edn. ARM Limited. http://www.altera.com/literature/third-party/archives/ddi0100e_arm_arm.pdf (2000). Accessed 15 July 2011
4. ARM Holdings profits up on tablet and smartphone sales. BBC News. http://www.bbc.co.uk/news/business-13207150 (2011). Accessed 15 July 2011
5. Blackman, S.S.: Multiple-Target Tracking with Radar Applications. Artech House Inc., Norwood, MA (1986)
6. Catsoulis, J.: Designing Embedded Hardware. O'Reilly (2005)
7. Clarke, P.: ARM reports sales, profits up in Q2. EE Times, News and Analysis. http://www.eetimes.com/electronics-news/4204942/ARM-reports-sales-profits-up (2010). Accessed 15 July 2011
8. Cong, J., Fan, Y., Han, G., Zhang, Z.: Application-Specific Instruction Generation for Configurable Processor Architectures. In: Proc. ACM International Symposium on Field-Programmable Gate Arrays, pp. 183–189 (2004). doi: 10.1.1.122.9578
9. DSP and SIMD. ARM Limited. http://www.arm.com/products/processors/technologies/dsp-simd.php. Accessed 15 July 2011
10. Dundamudi, S.P.: Guide to RISC Processors for Programmers and Engineers. Springer, United States (2004)
11. FPGAs for High-Performance DSP Applications. Altera Corporation. http://www.altera.com/literature/wp/wp_dsp_comp.pdf (2005). Accessed 11 July 2011
12. FPGAs Provide Reconfigurable DSP Solutions. Altera Corporation. http://www.altera.com/literature/wp/wp_dsp_fpga.pdf (2001). Accessed 11 July 2011
13. Ganssle, J., Barr, M.: Embedded Systems Dictionary. CMP Books (2003)
14. Goldston, J., Bhattacharya, R.: Reaping the Benefits of SoC processors for Video Applications. Texas Instruments. http://focus.ti.com.cn/cn/lit/wp/spry096/spry096.pdf (2007). Accessed 16 July 2011
15. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Pearson Prentice Hall, New Jersey (2008)
16. GSM Roaming and Coverage Maps. Mobile World Live. http://www.mobileworldlive.com/maps/ (2010). Accessed 14 July 2011
17. Gupta, V.K., Vinod, T., Gupta, K.: Compiler directed Customization of ASIP Cores. In: Proc. of the 10th Int'l Symp. on Hardware/Software, Codesign, pp. 97–102 (2002). doi: 10.1.1.16.8455
18. Heath, S.: Embedded Systems Design, 2nd edn. Newnes (2003)
19. Hu, j., Hoang X. D., Khalil, I.: An embedded DSP hardware encryption module for secure e-commerce transactions. In: Security and Communication Networks 4(8), 902–909 (2011). doi: 10.1002/sec.221
20. Islam, S., Ajmal, F.: Developing and implementing encryption algorithm for addressing GSM security issues. In: International Conference on Emerging Technologies, pp. 358–361 (2009). doi: 10.1109/ICET.2009.5353146
21. Katz, D.J., Gentile, R.: Memory Systems. In: Ganssle, J. (ed) Embedded Hardware, pp. 183–238. Newnes (2008)
22. Katz, D.J.: Embedded Media Processing (Embedded Technology). Newnes (2005)
23. Krishna, C.M., Shin, K. G.: Real-Time Systems. McGraw-Hill (1997)
24. Laplante, P.A.: Real-Time Systems Design and Analysis, 3rd edn. Wiley-IEEE Press (2004)
25. Lau, D., Blackburn, J., Seely, J.A.: The Use of Hardware Acceleration in SDR Wave-forms. Altera Corporation. http://www.altera.com/literature/cp/cp_sdr_hardware_acceleration.pdf (2005). Accessed 12 July 2011
26. Li, Q., Yao, C.: Real-Time Concepts for Embedded Systems. CMP Books (2003)
27. Liu, D.: Embedded DSP Processor Design: Application Specific Instruction Set Processors (Systems on Silicon). Morgan Kaufmann (2008)
28. Mody, M.: Video encoding, SoC development, and TI's DSP architecture. Texas Instruments. http://www.eetimes.com/design/signal-processing-dsp/4013053/Video-encoding-SoC-development-and-TI-s-DSP-architecture (2006). Accessed 15 July 2011
29. Noergaard, T.: Embedded Board Buses and I/O. In: Ganssle, J. (ed) Embedded Hard-ware, pp. 137–182. Newnes (2008)

30. Noergaard, T.: Embedded Processors. In: Ganssle, J. (ed) Embedded Hardware, pp. 63–136. Newnes (2008)
31. OMAP™ 2 Architecture: OMAP2420 Processor. Texas Instruments. http://focus.ti.com/pdfs/wtbu/TI_omap2420.pdf (2005). Accessed 17 July 2011
32. Oraioglu, A., Veidenbaum, A.: Guest Editors Introduction: Application Specific Microprocessors. IEEE Design & Test of Computers 20(1), 6–7 (2003). doi: 10.1109/MDT.2003.1173046
33. Oshana, R.: DSP Software Development Techniques for Embedded and Real-Time Systems. Newnes (2006)
34. Parker, M.: FPGA vs. DSP Design Reliability and Maintenance. Altera Corporation. http://www.altera.com/literature/wp/wp-01023.pdf (2007). Accessed 11 July 2011
35. Prasad, K.V.K.K.: Embedded/ Real-Time Systems: Concepts, Design and Programming. Dreamtech Press, New Delhi (2009)
36. Pratt, W.K.: Digital Image Processing: PIKS Scientific Inside, 4th edn. Wiley-Interscience, New Jersey (2007)
37. Proakis, J.G., Manolakis, D.G.: Digital Signal Processing: Principles, Algorithms and Applications, 4th edn. Prentice Hall (2007)
38. Quereshi, S.: Embedded Image Processing on the TMS320C6000™ DSP: Examples in Code Composer Studio™ and MATLAB. Springer, New York (2005)
39. Rabiner, L.R.: Digital Processing of Speech Signals. Prentice Hall (1978)
40. Sheldon, D., Kumar, R., Lysecky, R., Vahid, F., Tullsen, D.: Application-Specific Customization of Parameterized FPGA Soft-Core Processors. In: International Conference on, Computer-Aided Design (2007). doi: 10.1.1.76.3659
41. Sinha, P.: Speech Processing in Embedded Systems. Springer, New York (2010)
42. Smith, S.W.: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Pub. (1997)
43. Stankovic, J.: Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems. IEEE Computer 21(10), 10–19 (1988). doi:10.1109/2.7053
44. Stapko, T.: Practical Embedded Security: Building Secure Resource-Constrained Systems. Newnes (2007)
45. Stergiopoulos, S.: Advanced signal processing handbook: Theory and implementation for radar, sonar, and medical imaging real-time systems. CRC Press LLC (2001)
46. Stergiopoulos, S.: Implementation of adaptive and synthetic-aperture processing schemes in integrated active-passive sonar systems. Proc. IEEE. 86(2), 358–396 (1998). doi:10.1109/5.659491
47. The Evolving Role of FPGAs in DSP Applications. BDTI. http://www.bdti.com/MyBDTI/pubs/fpga_article.pdf (2007). Accessed 11 July 2011
48. Tretter, S.A.: Communication System Design Using DSP Algorithms: With Laboratory Experiments for the TMS320C6701 and TMS320C6711. Springer (2003)
49. Windrow, B., Stearns, S.D.: Adaptive Signal Processing. Prentice-Hall, Englewood Cliffs, NJ (1985)
50. Wolf, W.: Computers as Components: Principles of Embedded Computing System Design, 2nd ed. Morgan Kaufmann (2008)
51. Zhang, L., Li, S., Yin, Z., Zhao, W.: A Research on an ASIP Processing Element Architecture Suitable for FPGA Implementation. In: International Conference on Computer Science and, Software Engineering, 3, pp. 441–445, 2008. doi: 10.1109/CSSE.2008.580

http://www.springer.com/978-1-4614-7914-7

Secure Smart Embedded Devices, Platforms and Applications

Markantonakis, K.; Mayes, D.K. (Eds.)
2014, XLI, 568 p. 135 illus. in color., Hardcover
ISBN: 978-1-4614-7914-7