EURASIP Journal on Applied Signal Processing 2005:16, 2694–2700
© 2005 Hindawi Publishing Corporation

Multimedia Terminal System-on-Chip Design and Simulation

Ivano Barbieri
Department of Biophysical and Electronic Engineering, University of Genova, Via Opera Pia 11A, 16146 Genova, Italy
Email: ivano@dibe.unige.it

Massimo Bariani
Department of Biophysical and Electronic Engineering, University of Genova, Via Opera Pia11A, 16146 Genova, Italy
Email: bariani@dibe.unige.it

Alessandro Scotto
Department of Biophysical and Electronic Engineering, University of Genova, Via Opera Pia 11A, 16146 Genova, Italy
Email: scotto@dibe.unige.it

Marco Raggio
Department of Biophysical and Electronic Engineering, University of Genova, Via Opera Pia 11A, 16146 Genova, Italy
Email: raggio@dibe.unige.it

Received 30 January 2004; Revised 30 March 2005

This paper proposes a design approach based on integrated architectural and system-on-chip (SoC) simulations. The main idea is to have an efficient framework for the design and the evaluation of multimedia terminals, allowing a fast system simulation with a definable degree of accuracy. The design approach includes the simulation of very long instruction word (VLIW) digital signal processors (DSPs), the utilization of a device multiplexing the media streams, and the emulation of the real-time media acquisition. This methodology allows the evaluation of both the multimedia algorithm implementations and the hardware platform, giving feedback on the complete SoC including the interaction between modules and conflicts in accessing either the bus or shared resources. An instruction set architecture (ISA) simulator and an SoC simulation environment compose the integrated framework. In order to validate this approach, the evaluation of an audio-video multiprocessor terminal is presented, and the complete simulation test results are reported.

Keywords and phrases: system-on-chip, multimedia, HW-SW codesign, DSP, simulation, VLIW.

1. INTRODUCTION

Architecture achievements of the last years followed the HW-SW codesign approach transferring functionalities from hardware to software implementation [5] and moving developers toward system-on-chip (SoC) programmable devices. Otherwise SoC application-driven design seems to be the answer to fulfill multimedia applications requirements [5]. Moreover, thanks to the VLIW [6] architecture approach, it is now possible to design DSP-oriented chips that are stand-alone processors [7] reaching high degrees of parallelism [8]. Even though a number of general-purpose processors are suitable for DSP tasks, native DSP processors outperform general-purpose devices in terms of their cost-performance ratio and power consumption [9, 10].

In the following, a system-on-chip design based on a dual core DSP architecture will be described. The approach is based on HW-SW codesign, simulating at the same time the instruction set architecture (ISA) and the complete SoC, and taking into account single device performance together with run-time interactions between cores and peripheral devices.

2. MAIN ISSUE AND RELATED WORK

An HW-SW codesign environment is an application-driven tool chain able to evaluate and to modify both SW and HW at the same time. The software developer challenge is often to optimize a standard-based multimedia application [11, 12] to get real-time performance on a given architecture. The best result, following a given design constraints list, is achievable by tuning both the algorithm implementation and architectural features. Instruction simulators are nowadays widely used in developing application-driven architecture design. Architecture simulation incorporates several simulation techniques ranging from interpretive simulation to application-architecture-specific compiled simulation techniques. Even though the compiled simulation approach allows better peak performance in terms of simulated instructions per second, interpretive simulation appears to be more flexible [5]. Most of the state-of-the-art multimedia and DSP device solutions are based on SoC. The development environment should allow simulating run-time interactions between core and peripheral devices in an SoC. In this scenario, the interpretive approach best matches heterogeneous devices simulation, especially regarding interdevice run-time interactions, which are not easily predictable a priori. An example of an SoC simulation environment using an ISS integration approach is introduced in [13]. The proposed solution uses cycle-accurate ISSs integrated with the architecture design environment. The work in [13] also analyses tradeoffs between ISS and communication interfaces in SoC simulation environments and highlights the importance of architectural visibility and of debug interface. The work in [14] presents an innovative approach for the design of application specific multiprocessor SoC. Using a generic architecture model as initial template, available resources are divided in software, hardware, and communication components. In the SoC validation process, CPUs are replaced by cycle-accurate ISSs. However, cycle-accuracy often leads to low performance, whereas a vertical optimisation for improving simulation speed will have a negative effect on flexibility. Usually simulation environments try to achieve the best compromise between speed and accuracy depending on the scope of the development environment.

The methodology introduced in this paper is based on both architecture and SoC simulation. The presented approach employs an interpretative ISS integrated in an SoC design tool for fast simulation in order to obtain an effective architectural exploration and to investigate new SoC solutions. Compared to other existent hardware design tools, this methodology allows to obtain efficient system simulations in a short time with parametrical accuracy and good observability.

An efficient HW-SW codesign development environment should also allow continuous access to registers and bus values during the simulation providing statistics about the use of the different resources [11]. The capability to collect statistics on both system devices, for example, internal resource utilization, and interdevice communications allows a software developer to better focus his work either on a particular application or on a portion of the selected code.

In this paper we demonstrate the effectiveness of using integrated architectural and SoC design environments, reducing the time to identify major design constraints and bottlenecks starting from the software application development. The described methodology does not limit hardware accuracy, allowing to perform subsequent steps of the system design.

The proposed SoC solution is mainly based on two VLIW cores working in parallel, a multiplexer, and a bus arbiter. The multiplexer component implements the H.223 standard [15]. H.223 has a flexible mapping scheme that is suitable for a variety of media and can handle variable frame lengths. This solution is independent from streaming applications allowing any composition of different media channels (audio/video/data communication control) and implements different protocols, namely H.223 annexes A and B [15], depending on the type of multimedia terminal, such as wired PSTN-ISDN [16], and wireless 3GPP [17].

The H.223 flexibility and the utilization of programmable DSPs together with the definition of a generic communication interface that controls the devices' behaviour and the interactions among the SoC modules, allow the design and evaluation of multimedia terminals using different streaming standards.

The chosen SoC scheme, modules, and HW-SW partition, including two general purpose cores, one mux-specific-device, and memories, have been selected to address the streaming process for a given family of multimedia terminals consisting of ITU-T H.32x and 3GPP standards. SoC scheme parameters are, for example, memories size, bus width, and multiplexer configuration. From the signal processing point of view, the scheme supports several compression algorithms such as H.263, MPEG4, G.723, and AMR using, as described above, different parameters.

The organization of the paper is as follows. In Section 3, the SoC modelling the multimedia terminal is described. This section details the most important devices composing the overall system; attention is focused on main features and the communication interface. The proposed approach has been validated on an audio-video multiprocessor system reported in Section 4.

Simulation test results for this case study are also reported in Section 4. Finally, conclusions are drawn in Section 5.

3. SYSTEM-ON-CHIP DESCRIPTION

The SoC includes two processors implementing audio and video compressions. Each processor is connected to three external memories: a program memory, a data memory, and a memory containing the audio or video data to process. The third memory is loaded at the reset phase with the proper data to process and it is utilized, together with a timer, in order to simulate the real-time acquisition (audio and video) behaviour.

Audio and video memories are loaded at the simulation reset-phase starting from a specified address. The core processors load audio/video data according to the real-time acquisition timers and process the input data producing compressed streams to be sent to the multiplexer (MUX). The audio coder algorithm synchronizes the production of the multimedia terminal output stream. Therefore the audio processor generates an interrupt at the end of each audio-frame processing, signaling to the multiplexer that the data are available to compose a new MUX-frame.

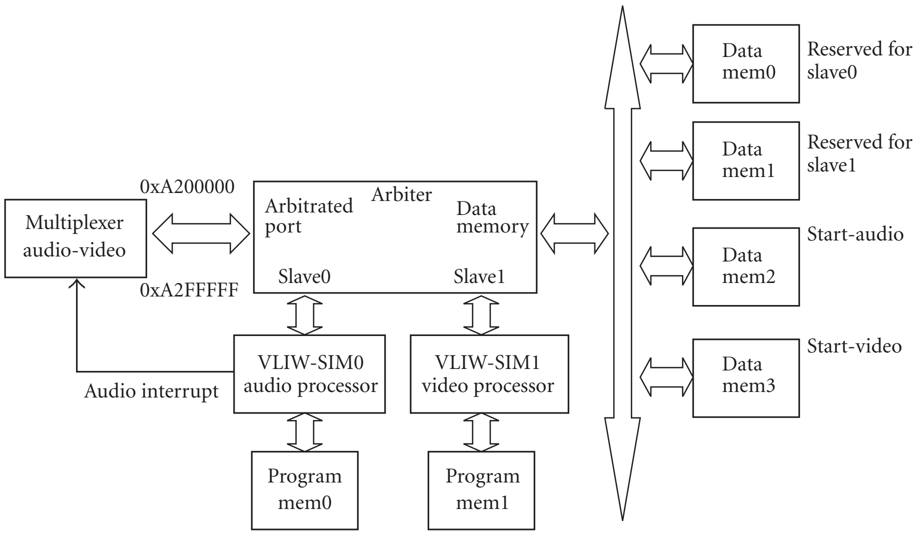

Figure 1: Audio-video system.

The multiplexer put-together the available data to form an audio-video packet according to the H.223 standard syntax and writes the produced stream to an output file.

The communication between processors and memories occurs through a bus arbiter, that also regulates the accesses to the shared-resource multiplexer.

The SoC architecture has been designed and simulated using the MaxSim environment. MaxSim is a simulation environment for easy modelling and fast simulation of integrated SoC with multiple cores, peripherals, and memories. The cycle-based scheduler and the transaction-based component interfaces enable high simulation speed while retaining full accuracy. The MaxSim system can be used as a standalone simulation module or integrated into HW simulation or HW/SW cosimulation tools [3].

The whole system simulation occurs without any I/O operations except for the storing of the MUX stream to the output file. Figure 1 shows devices and connections in the simulation. The modules composing the proposed SoC are detailed in the rest of this section.

3.1. Bus arbiter

This module regulates all the communications between master (core processors) and slave devices (e.g., memories, MUX) in the SoC. The *bus arbiter* receives the read-from/write-to memory requests from the two masters and sends them to the appropriate slave device depending on the address range. It also resolves simultaneous access requests to shared resources. The bus arbiter interface includes two ports connected to master devices and six ports connected to slave devices. The two data memories *datamem0* and *datamem1* are reserved for the master devices 0 and 1, while other memory devices (*datamem2*, *datamem3*) can be used without constraints. The port named *arbitratedport* is reserved for connecting the shared device. In case of simultaneous access to shared devices, the bus arbiter grants the access to the higher-priority master device. In this project, the higher-priority master is the processor device simulating the audio compression (device 0). Data in memories are stored starting from the address 0x00000000, therefore the bus arbiter uses an offset part of a data address to identify the memory-device and the displacement to find the internal memory address. During the simulation phase, the bus arbiter collects statistics on both read and write accesses from master devices, moreover it measures the number of contemporary shared-resource request (conflicts).

3.2. Multiplexer

The multiplexer receives audio and video data and puts them together in a MUX-frame packet following an H.223-like syntax. Each MUX-frame consists of a header followed by audio and video data. The header is composed of the following.

(i) Start of frame: two bytes usable as CRCa, and so forth.
(ii) MUX table index: indicates the MUX-frame composition (audio and/or video).
(iii) MUX-frame size: indicates the total size in bytes of the MUX-frame.

The number of the audio bytes in a MUX-frame is a multiplexer parameter, varying depending on the audio compression algorithm; it is a fixed size for each MUX-frame. The number of video bytes in a MUX-frame is a variable value influencing the MUX-frame total size; it can vary among MUX frames up to the maximum MUX-frame size and is therefore limited by the physical channel bandwidth. The audio processor synchronizes the multiplexer generating an interrupt at the end of each audio processing, signaling that the data are ready for composing a new MUX-frame packet. In order to continuously process data, the multiplexer uses two ping-pong buffers. When an audio interrupt is received, the ping-pong buffers are swapped allowing the multiplexer to assemble the data from header, audio, and video components while audio and video processors keep sending data. Since the ping-pong swapping of the buffers is controlled by the audio interrupt, a delay in audio processing would result in an overflow of the video buffers. In order to avoid the video buffer overflow, the ping-pong swapping is also activated whenever the video buffer size limit is reached.

3.3. The VlIW-SIM component

The VLIW DSP cores are simulated using the VLIW-SIM ISA simulator. VLIW-SIM is an interpretative reconfigurable ISA simulation environment supporting both application design and architectural exploration. A dynamic library version of the simulator has been developed in order to utilize VLIW-SIM not only as stand-alone core simulator but also as a component in a more complex HW-SW codesign environment. The ISS has been developed in pure C-language in order to reach high simulation speed [1].

The VLIW-SIM component offers all the functionalities of the stand-alone version, such as resource observability and application profiling, allowing to change the simulated target processor to maintain the same component interface without affecting the other modules of which the SoC comprises.

The interface between VLIW-SIM and other devices is implemented through communication ports. The read/write operation on a data memory port is divided into two phases: *access request* and *check for grant*. The VLIW-SIM component is able to either stall, for example, when the bus controller does not grant the access for a read or write operation, or generate an interrupt to signal a particular event to other SoC modules.

The data to be processed by the coding applications are stored in data memory starting from a specific address depending on the media type. The coders process one frame and store the processed data to the output buffer at the proper MUX-field starting address. As explained in Section 3.2, the multiplexer waits for the audio interrupt signal to generate the MUX-frame packet and to swap ping-pong buffers for receiving the successive compressed frames.

To better simulate the real-time acquisition, a timer is used in order to start the compression of each frame.

4. A CASE STUDY

The presented design approach is here utilized to design and evaluate a multiprocessor system for audio-video compression. The simulated multimedia terminal is based on two ST210 [18] processors working in parallel, one dedicated to the compression of a video stream following the ITU-T H.263 [19] standard protocol, and the other one executing the ITU-T G.723 [20] speech compression. Data generated by the two processors are multiplexed together in an H.223-like [15] format. The selected applications belong to a set of multimedia algorithms preventively defined as relevant application to verify SoC performance and code optimization. The reported tests have been performed on optimized implementations of the ITU-T standard algorithms specifically implemented for the addressed target architecture.

The VLIW-SIM simulator has been configured to simulate the following ST210 architecture.

(i) Clock rate: 250 MHz.
(ii) I-cache: 32 kB, directly mapped.
(iii) D-cache: 32 kB, 4-way associative (round-robin block replacement).

The real-time acquisition constraints are taken into account by timers since the G.723 standard expects to process an audio frame every 30 millisecond. The produced output stream has been decoded using a standard A/V terminal decoder in order to check stream compatibility.

Figure 2 shows the MaxSim project for the described system. It is composed of two ST210 cores and their correspondent program memories, four data memories (two for audio processing and two for video processing), two devices for audio and video memories initialization (called *memloaders*), a bus arbiter, a multiplexer, and two real-time acquisition timers.

The bus arbiter measures audio and video processor accesses for read and write operations taking also into account simultaneous multiplexer accesses and stalls.

Table 1 shows the collected statistics about the processors and the bus arbiter. Tests described in this document have been performed using the following platform.

(i) OS: Win 2000, service pack 3.
(ii) CPU: Pentium IV 2.0 GHz.
(iii) RAM: 256 MB.

The *cycles* entry in Table 1 represents the number of clock cycles without stalls due to cache miss, which for VLIW architectures is the number of executed *long instructions*. *Operations* is the number of elementary instructions executed in the simulated application except for NOP instructions, a number of instructions ranging from one to the long instruction size are executed in parallel for each cycle. *D-cache-misses* is the number of write and read misses in data-cache, while *I-cache-misses* represents the number of read misses in instruction-cache. *Data-mem-accesses* is the whole number of accesses in data memory including both D-cache hits and misses, while the whole number of accesses in program memory comprising both I-cache hits and misses is denoted by *instr-mem-accesses*. *Bus accesses on write* is the total number of accesses from master devices to bus for write operations. *Bus accesses denied* is the total number of conflicts in accessing the shared device (MUX). Finally *simulation time* is the time elapsed to complete the application simulation. The latter measures the ISA simulator performance, and allows, together with *operation,* to calculate the number of simulated cycles per second.

In our test, processing 28 audio frames and 20 video frames, multiplexer access conflicts do not occur. This is mainly due to the difference in read/write operation frequency of the two algorithms giving a probability of a contemporary access to the multiplexer near to zero.

The H.263 and G.723 encoders used in this test are ST210 optimized implementations, allowing a frame compression

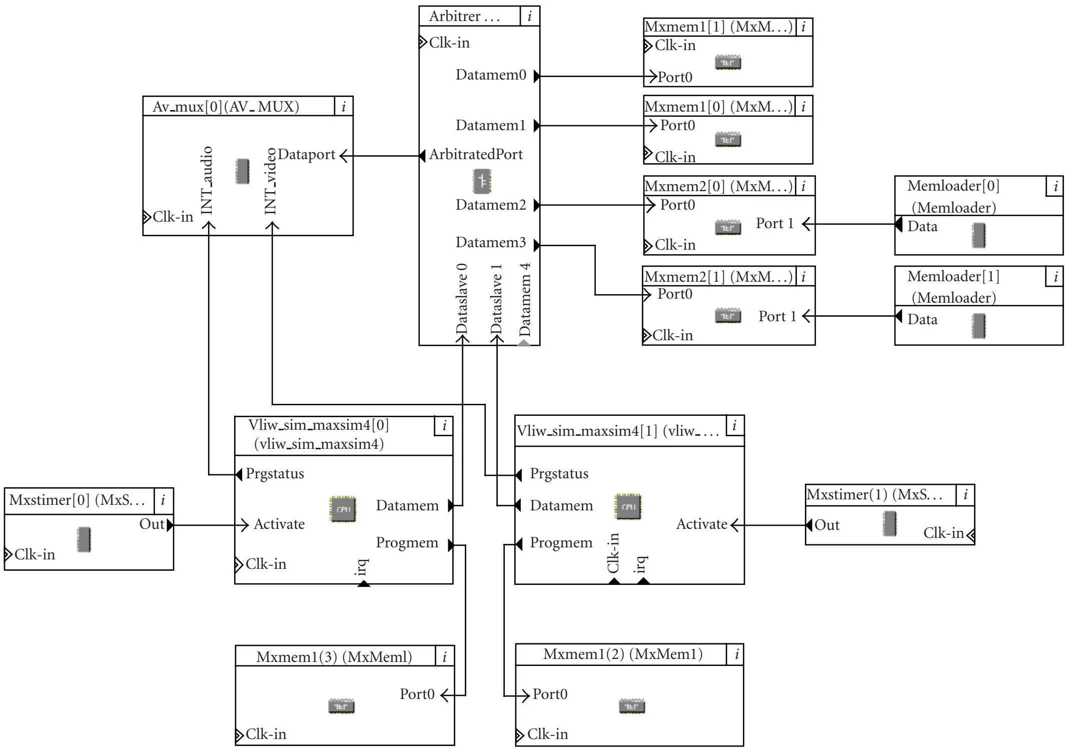

Figure 2: Audio-video system.

Table 1: Simulation statistics.

	VLIW-SIM 1 Video	VLIW-SIM 0 Audio
Cycles	210 002 493	210 002 870
Stalls	156 695 902	194 281 104
(% total cycles)	(74.61%)	(92.51%)
Operations	132 141 925	43 936 322
D-cache misses	310 074	1771
I-cache misses	199 401	28 479
Data mem accesses	25 535 279	8 122 811
Instr mem accesses	144 536 244	46 094 608
Bus accesses on read	18 023 276	6 865 049
Bus accesses on write	7 512 444	1 258 055
Bus accesses denied	0	0
Simulation time (s)	296	296
Simulated cycle/s	–	709 310

time considerably below the real-time temporal window. Therefore, ST210s are stalling for most of the processing time (video: 74.61%, audio: 92.51%), waiting for real-time data acquisition. This result indicates, for example, the possibility to decrease ST210s clock rates in order to satisfy power consumption constraints, or to select a different architecture for the media processing. Otherwise, keeping the same HW configuration, decoders, and other terminal communication modules such as H.245 control [21], speech recognition module, or video acquisition processing modules as RGB to YCbCr conversions, can be implemented on the ST210 cores. Alternatively, a multichannel system can be developed executing multiple coder instances on each core.

5. CONCLUSIONS

In this paper we have presented a design methodology based on an ISA simulator integrated into an SoC design environment. Differently from other existent HW-design tools, this methodology allows to have an effective system simulation in a short time with parametrical degrees of accuracy and observability. Moreover, the described environment allows a "system profiling," both detecting bottlenecks in communication between system components and evaluating the degree of optimization in the simulated applications using system criteria instead of single-processor criteria. The simulation speed and the short implementation time for each block comprising of the creation of MaxSim components and the reconfiguration of VLIW-SIM allow us to develop the SW applications algorithms together with the HW system architectures.

In order to validate this methodology, the case study of a multiprocessor system for audio-video compression and multiplexing has been presented. In this SoC design evaluation each block is reusable and allows internal architecture exploration to model the different HW behaviors. Future work will focus on power estimation models connected with core simulation oriented description in order to take into account power efficiency in both hardware design and application development.

REFERENCES

[1] I. Barbieri, M. Bariani, A. Cabitto, and M. Raggio, "Multimedia-application-driven instruction set architecture simulation," in *Proc. IEEE International Conference on Multimedia and Expo (ICME '02)*, vol. 2, pp. 169–172, Lusanne, Switzerland, August 2002.

[2] I. Barbieri, M. Bariani, A. Cabitto, and M. Raggio, "Efficient DSP simulation for multimedia applications on VLIW architectures," in *Mathematics and Simulation with Biological Economical and Musicoacoustical Applications*, pp. 83–86, Malta, September 2001.

[3] AXYS® Design Automation Inc, http://www.axysdesign.com/.

[4] AXYS Design Automation Inc, "MaxSim Developer Suite User's Guide Version 3.0," Document Version 1.04 September 9th, 2002.

[5] A. Hoffmann, T. Kogel, A. Nohl, et al., "A novel methodology for the design of application-specific instruction-set processors (ASIPs) using a machine description language," *IEEE Trans. Computer-Aided Design*, vol. 20, no. 11, pp. 1338–1354, 2001.

[6] J. A. Fisher, "Very long instruction word architectures and the ELI-512," in *Proc. 10th Annual International Symposium on Computer Architecture (ISCA '83)*, pp. 140–150, Stockholm, Sweden, June 1983.

[7] P. Faraboschi, G. Desoli, and J. A. Fisher, "The latest word in digital and media processing," *IEEE Signal Processing Mag.*, vol. 15, no. 2, pp. 59–85, 1998.

[8] B. R. Rau and J. A. Fisher, "Instruction-level parallel processing: history, overview and perspective," *Journal of Supercomputing*, vol. 7, no. 1-2, pp. 9–50, 1993, Special issue on instruction-level parallelism.

[9] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals: Architectures and Features*, IEEE Press, Piscataway, NJ, USA, 1996.

[10] J. Bier, "VLIW architectures for DSP: a two-part lecture outline," in *Proc. International Conference on Signal Processing Application and Techniques (ICSPAT '99)*, pp. 1290–1301, Orlando, Fla, USA, November 1999.

[11] A. Jemai, P. Kission, and A. A. Jerraya, "Architectural simulation in the context of behavioral synthesis," in *Proc. Conference on Design, Automation and Test in Europe (DATE '98)*, pp. 590–595, Paris, France, February 1998.

[12] A. Jemai, P. Kission, and A. A. Jerraya, "Embedded architectural simulation within behavioral synthesis environment," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC '97)*, pp. 227–232, Makuhari Messe, Chiba, Japan, January 1997.

[13] L. Guerra, J. Fitzner, D. Talukdar, C. Schläger, B. Tabbara, and V. Zivojnovic, "Cycle and phase accurate DSP modeling and integration for HW/SW co-verification," in *Proc. 36th Design Automation Conference (DAC '99)*, vol. 1, pp. 964–969, New Orleans, La, USA, June 1999.

[14] A. Baghdadi, D. Lyonnard, N. E. Zergainoh, and A. A. Jerraya, "An efficient architecture model for systematic design of application-specific multiprocessor SoC," in *Proc. Conference on Design, Automation and Test in Europe (DATE '01)*, pp. 55–62, Munich, Germany, March 2001.

[15] ITU-T Recommendation H.223, "Multiplexing protocol for low bitrate multimedia communication," May 1995.

[16] ITU-T Recommendation H.324, "Terminal for low bit-rate multimedia communication," March 2002.

[17] 3GPP specification TS 26.111, "Technical Specification Group Services and System Aspects; Codec for circuit switched multimedia telephony service; Modifications to H.324," June 2003.

[18] ST200 Core Architecture Manual, STMicroelectronics 2000.

[19] ITU-T Recommendation H.263, "Video coding for low bitrate communication," February 1998.

[20] ITU-T Recommendation G.723.1, "Dual rate speech coder for multimedia communication transmitting at 5.3 and 6.3 kbit/s," March 1998.

[21] ITU-T Recommendation H.245, "Control protocol for multimedia communication," July 2003.

Ivano Barbieri was born in Genova, Italy, in 1969. He obtained his M.S. degree in electronic engineering from Genoa University—thesis on "Research on image quality evaluation alternative methods to MSE (mean square error) in image coding systems for the subjective redundancy reduction"—and Ph.D. degree—thesis on "Efficient methodologies for multimedia communication terminal design and testing." Since 1995, he has been employed in the Department of Biophysical and Ellectronic Engineering (DIBE), Genoa University. Research areas are innovative approaches on image quality evaluation, architectural research on systems for real-time efficient implementation of video coding algorithms exploring both embedded and single-chip solutions, real-time multimedia systems (platforms, multiplexing, and control issues), DSP architecture and development environments, architecture modelling for media processing, and embedded systems for mobile (low-power) applications.

Massimo Bariani was born in Genova, Italy, in 1970. He obtained his M.S. degree in electronic engineering from Genoa University—thesis on "Development and implementation of a multipoint control unit for multimedia videoconferences"—and Ph.D. degree—thesis on "Modelling and simulation of VLIW architectures for HW-SW codesign of embedded systems." He is currently employed as a researcher in the Department of Biophysical and Electronic Engineering (DIBE), Genoa University. In the electronic system field, his interests include hardware design and simulation, multimedia algorithm implementation for VLIW architectures, architectural exploration, and real-time multimedia communication based on standard protocols.

Alessandro Scotto was born in Genova, Italy, in 1976. He obtained his M.S. degree in electronic engineering from Genoa University in 2001—thesis on "Multichannel system for real-time voice coding on DSP architecture. Since 2002, he has been a Ph.D. student in the Department of Biophysical and Electronic Engineering (DIBE), Genoa University. Research areas are architectural research on systems for real-time efficient implementation of video-voice coding algorithms exploring both embedded and single-chip solutions, DSP architecture and development environments, architecture modelling for media processing, and embedded systems for mobile (low-power) applications.

Marco Raggio was born in Chiavari, Genova, Italy, in 1964. He obtained his M.S. degree in electronic engineering from Genoa University—thesis on "Development and real-time test of video compression algorithms"—and Ph.D. degree—thesis on "Implementation and simulation of real-time multimedia embedded system for videotelephony application and advanced DSP architecture. "Since 1995, he has been employed as a Research Project Manager/Officer in the Department of Biophysical and Electronic Engineering (DIBE), Genoa University. In the electronic system field, his interests involve hardware design and simulation, interactive real-time multimedia architecture design, that is, for mobile terminal and surveillance systems. His activities also involve field trials setup, audit, and dissemination. In the networking field, he has expertise in LAN design, configuration, maintenance, and security. He participates in on university seminars with discussions on video coding, standards for multimedia and streaming, DSP, industrial field bus, and embedded systems.