

AUDIO-VIDEO TERMINAL SYSTEM-ON-CHIP SIMULATION

Ivano Barbieri, Massimo Bariani, Marco Raggio, Alessandro Scotto

Biophysical and Electronic Engineering Department, University of Genoa
Via Opera Pia 11 A, 16146 Genoa ITALY
phone: +39 010 3532 273, fax: +39 010 3532 036, email: ivano@dibe.unige.it
web: <http://www.esng.dibe.unige.it/Staff/IvanoBarbieri.html>

ABSTRACT

In this paper, a system-on-chip design and simulation of an Audio-Video encoding terminal has been described. The purpose of this work was to model a multimedia mobile terminal in order to preliminarily explore system bottlenecks and inter-device communications. The system is based on two ST210[1] processors working in parallel, one dedicated to the compression of a video stream following the ITU-T H.263 [2] standard protocol, and the other one executing the ITU-T G.723 [3] speech compression. Data generated by the two processors are multiplexed together in an H.223-like [4] format. The ST210 cores are simulated using the VLIW-SIM[5][6] environment targeted for ST210 architecture. VLIW-SIM is a retargettable Instruction Set Simulator (ISS), pipeline and cycle accurate able to model state of the art VLIW (Very Long Instruction Word) architectures. The complete System has been simulated using the MaxSim SoC simulation environment[7][8]. Simulation test results for the complete system are reported.

1. INTRODUCTION

Architecture achievements of the last years followed the Hw-Sw co-design approach transferring functionalities from Hardware to Software implementation [9] and moving developers toward system-on-chip programmable devices. Otherwise system-on-chip application driven design seems to be the answer to fulfill Multimedia applications requirements [9]. Moreover, thanks to VLIW (Very Long Instruction Word)[10] architecture approach, it is now possible to design DSP-oriented chips that are stand-alone processors [11] reaching high parallelism degrees (Instruction Level Parallelism - ILP) [12]. Even though a number of general-purpose processors are suitable for DSP task, native DSP processors outperform general-purpose in cost-performance rate and power consumption [13] [14].

In the following chapters a system-on-chip, based on a dual core DSP architecture has been described. The approach is based on Hw-Sw co-design, simulating at the same time the Instruction Set Architecture (ISA) and the complete system-on-chip, taking into account single device performance together with run time interactions between cores and peripheral devices.

2. SOC DESCRIPTION

The system simulates a dual-processor system-on-chip (SoC) 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) behavior. The SoC architecture has been designed and simulated using the MaxSim environment. MaxSim is a simulation environment for easy modeling 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 systems can be used as standalone simulation models or integrated into HW simulation or HW/SW co-simulation tools [7].

Audio and video data are loaded at the simulation-reset phase starting from a specified memory address. The communication between processors and memories occurs through a bus managing memories and other shared resources accesses depending on address range. Moreover, the bus regulates accesses to the multiplexer, whose task is to collect together received compressed data from audio and video encoders.

The audio processor controls the multiplexer through an interrupt signal. The interrupt is generated when the encoding of an audio frame has been completed. When the multiplexer (MUX) receives an interrupt signal, compressed audio data together with all the video data received since the previous audio interrupt, are collected together creating a MUX frame subsequently stored in an output file. MUX is a shared resource, therefore the bus detect and resolve all the contemporary accesses to this device.

The whole system simulation occurs without any I/O operations except for the storing of the MUX stream in the output file. Figure 1 shows devices and connections in the simulation.

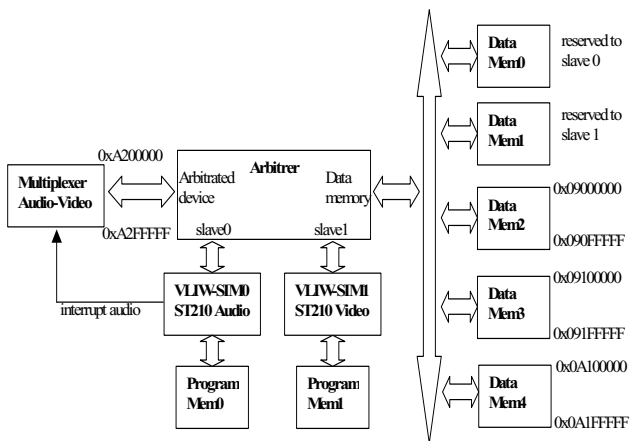


Figure 1 Audio-Video System

3. SOC DEVICES

3.1 Bus Arbiter

The *Bus Arbiter* regulates communications between master (ST210 processors) and slave devices (e.g. memories, mux) in the SoC. The *Bus Arbiter* receives the read-from/write-to memory address requests from the two master and send 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*, and *DataMem4*) can be used without constrains. 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 use 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

Multiplexer receives audio and video data and put them together in a Mux-frame packet following an H.223-like syntax. Each mux-frame is composed by a header respectively followed by audio and video data. The header is composed by:

- Start of Frame -two bytes usable as CRC etc.
- Mux Table Index – indicates the mux-frame composition (audio and/or video)
- Mux-frame Size – indicates the total size in bytes of the mux-frame

The number of audio bytes in a mux-frame is a multiplexer parameter, varying depending on the audio compression algorithm (audio bitrate); it is a fixed size for each mux-frame. The number of video bytes in a mux-frame is a parameter influencing the mux-frame total size; it can vary between mux-frames up to the maximum mux-frame size. The Mux-frame maximum size depends on the physical channel bandwidth. The audio processor (ST210 compressing audio) synchronizes the multiplexer generating an interrupt at the end of each audio processing, signaling that data are ready for composing a new mux-frame packet. In order to continuously process data, multiplexer uses two ping-pong buffers. When an audio interrupt is received, the ping-pong buffers are swapped allowing multiplexer to put received data together (header + audio + video) while audio and video processors keep on sending data. Moreover the multiplexer detect the End-of-Sdu symbol [4] in video stream data. The video coding introduces an End-of Sdu symbol composed by two bytes (255, 0) at the end of each video frame. To avoid the End-of-Sdu symbol emulation in the video stream, a byte-stuffing mechanism has been implemented. The MUX implements reverse byte-stuffing and takes care of audio-video synchronization. In the case of End-of Sdu symbol detection before audio interrupt, the subsequent video stream is stored in the other ping-pong buffer in order to be inserted in the next mux-frame. Since buffer ping-pong swapping is usually controlled by audio interrupt, a delay in audio processing would overflow video buffers. To avoid video buffer overflow, the ping-pong swapping is activated also when video buffer size limit is reached.

3.3 The VLIW-SIM component

The SoC main processing (audio and video compressions) is performed by two ST210 cores. Each core has been simulated using the VLIW-SIM Instruction Set Architecture (ISA) simulator. VLIW-SIM is an interpretative re-configurable ISA simulation environment supporting both application design and architecture 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 SOC Hw-Sw co-design environment. The ISS has been developed in pure C-language in order to reach high simulation speed and to be oriented to programmers without deep hardware knowledge of CPU architectures [5].

The interface between VLIW-SIM and other devices is implemented through communication ports. The read/write operation on data memory port is divided in two phases: *Access request* and *Check for grant*. Bus controller masters read and write operations. If the Bus does not grant the access, the processor stalls attempting to access memory in the next cycle. If the simulated application addresses an external device port, an interrupt is generated on the PrgStatus pin of the VLIW-SIM (ST210) processor.

VLIW-SIM has been configured to simulated the following ST210 architecture:

- Clock Rate: 250 MHz
- Single Cluster
- I-Cache type: Direct Mapped.
- I-Cache size: 32 Kbytes.
- I-Cache line size: 64Byte.
- D-Cache type: 4-way Associative (Round-Robin block replacement).
- D-Cache size: 32 Kbytes.
- D-Cache line size: 32 Bytes.

4. AUDIO AND VIDEO APPLICATIONS

The selected applications belong to a set of multimedia algorithms preventively defined as relevant application to verify SoC performance and code optimization. Tests have been performed on optimized implementations of ITU-T standard algorithms: the H.263+ video coder [2], the G.723.1 speech coder [3], specifically implemented for the addressed target architecture. Audio and video application parameters are described in the following:

H.263 coder

- Test Sequence: foreman.yuv
- Number of frame: 20(from frame 0 (Intra) to frame 19)
- Video input format: QCIF (176 x 144 pixel)
- Quantization index for P-frame: 10
- Quantization index for I-frame: 10
- Rate-control algorithm: None
- Motion estimation search algorithm: Improved Gradient Descent Search
- Motion estimation search window: 15
- Half Pixel Motion Estimation type: subset half pixels

G.723.1 coder:

- Test Sequence: ITU standard
- Number of encoded frames: 28 (from frame 0 to frame 27)
- Audio input format: 8 KHz - 16 bit
- Output Rate: 5.3 Kbps

4.1 G.723 audio coder

In the audio coding application, data to be processed are stored in memory starting from the START_AUDIO address. G.723.1 compress audio frame and stores processed data in the output buffer at the audio mux-field starting address. After this operation, an interrupt from the (audio) ST210 to the multiplexer is generated. The multiplexer swap ping pong buffers and generate the mux-frame packet. To better simulate the real-time audio acquisition, a timer is used in order to start the audio compression for each audio sample (every 30 ms).

4.2 H.263 video coder

In the video coding application, data to be processed are stored in memory starting from the START_VIDEO address. H.263+ compress video frame and stores processed

data in the output buffer at the video mux-field starting address. Multiplexer receive these data storing them in internal buffers taking into account the variable number of received video bytes. Like for the audio acquisition, a timer is used for simulating the real-time video acquisition.

5. TEST RESULTS

In the following tests the designed system-on-chip has been simulated in processing standard video and audio sequences and multiplexing compressed stream in order to keep audio/video synchronization. Real acquisition constraints are also taken into account with timers. The output stream has been decoded using a standard A/V terminal decoder in order to check stream compatibility. Figure 2 shows the Max-Sim project for the described system. It is composed by:

- Two VLIW-SIM processors (VLIW-SIM-ST210 simulation dlls) and its correspondent program memories.
- Four Data memories, two for audio processing and two for video processing.
- MemLoaders devices for Audio and Video memories initialization
- The Bus Arbiter
- The multiplexer
- Audio and Video acquisition timers

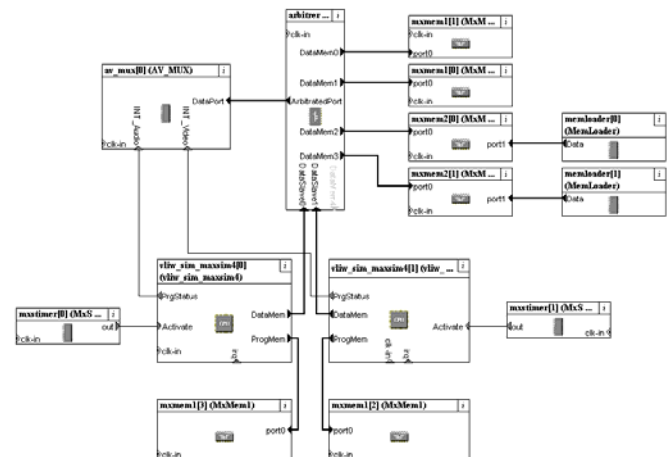


Figure 2 Audio-Video System

The Bus Arbiter measures audio and video processors accesses for read and write operation 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:

- O.S.: Win 2000, Service Pack 3 / Linux Red Hat 7.3
- CPU: Pentium IV 2.0 GHz
- RAM: 256 MB

	VLIW-SIM 1 Video	VLIW-SIM 0 Audio
Bundles	210002493	210002870
Stalls (% total cycles)	156695902 (74,61%)	194281104 (92,51%)
Operations	132141925	43936322
D-Cache Misses	310074	1771
I-Cache Misses	199401	28479
Data Mem Ac- cesses	25535279	8122811
Instr Mem Ac- cesses	144536244	46094608
Bus Accesses on Read	18023276	6865049
Bus Accesses on Write	7512444	1258055
Bus Accesses Denied	0	0
Simulation time [sec]	296	296
Simulated Cy- cle/Sec	709310	

Table 1 Simulation statistics

In our tests 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.

H.263 and G.723 encoders used in this test are ST210 optimized implementation, allowing compression-frame time highly below the real-time temporal window. Therefore, ST210s are stalling for most of the processing time (video: 74,61%, audio: 92,51), waiting for data from acquisition. This results indicates, for example, the possibility to decrease ST210s clock rates in order to satisfy power consumption constraints or alternatively to execute multiple coder instance on each ST210 for implementing a multi-channel system.

6. CONCLUSIONS

In this paper we presented the first step in SoC design evaluation of a multiprocessor system for audio-video compression and multiplexing. Each single block is re-usable and allows internal architecture exploration to model the different HW behaviors.

Both the simulation speed and the short implementation time for each block (MaxSim components and VLIW-SIM) allow developing the Sw applications (algorithms) together with the Hw system (architectures).

Differently from other existent Hw-Design tool, 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 de-

gree of optimization in the simulated applications using system criteria instead of single-processor criteria. 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] ST200 Core Architecture Manual, STMicroelectronics 2000
- [2] ITU-T Recommendation H.263, "Video coding for low bitrate communication", February 1998
- [3] ITU-T Recommendation G.723.1, "Dual rate speech coder for multimedia communication transmitting at 5.3 and 6.3 kbit/s", March 1998
- [4] ITU-T Recommendation H.223, "Multiplexing protocol for low bitrate multimedia communication", May 1995.
- [5] I. Barbieri, M. Bariani, M. Raggio, "Multimedia Application Driven Instruction Set Architecture Simulation" ICME2002 - International Conference on Multimedia and Expo 2002 - Aug 2002, Lausanne
- [6] I. Barbieri, M. Bariani, A. Cabitto, M. Raggio, "Efficient DSP simulation for Multimedia Applications on VLIW architectures" in "Mathematics and simulation with biological economical and musicoacoustical Applications", pp. 83-86, Ed. by Attelis-Kluev-Mastorakis 2001 - WSES press - ISBN 960-8052-46-7
- [7] AXYS® Design Automation, Inc, <http://www.axysdesign.com/>
- [8] AXYS Design Automation Inc, "MaxSim Developer Suite User's Guide Version 3.0", Document Version 1.04 September 9 th , 2002
- [9] A. Hoffmann, T. Kogel, A. Nohl, G. Braun, O. Schliebusch, O. Wahlen, A. Wiefierink, H. Meyr, "A Novel Methodology for the Design of Application-Specific Instruction-Set Processors (ASIPs) Using a Machine Description Language", IEEE Transaction on Computer-Aided Design of Integrated Circuits and System, Vol. 20, N. 11, November 2001
- [10] Joseph A. Fisher. "Very long instruction word architectures and the ELI-512", Proceedings of the 10th Annual International Symposium on Computer Architecture, Stockholm, Sweden, June 1983.
- [11] P. Faraboschi, G. Desoli, J.A. Fisher "The Latest Word in Digital and Media Processing" IEEE Signal Processing Magazine, March 1998
- [12] B.R. Rau, J.A. Fisher. "Instruction Level Parallelism" The Journal of Supercomputing 7 May 1993
- [13] P. Lapsley, J. Bier, A. Shoham, E.A. Lee, "DSP Processor Fundamentals: Architectures and Features", IEEE Press series on Signal Processing, 1996.
- [14] Berkeley Design Technology Inc, "VLIW Architectures for DSP", DSP World/ICSPAT, Orlando Florida, November 1999