HARDWARE PROTOTYPING OF ARM BASED SYSTEM ON CHIP

Ivan Petkov, Marcio Oyamada, Paul Amblard, Ahmed Jerraya, Marin Hristov

ECAD Laboratory, 8 bul. Kliment Ohridksi - 1000 Sofia – Bulgaria
TIMA Laboratory, 46 av. Felix Viallet - 38031 Grenoble – France
e-mail : petkov@ecad.tu-sofia.bg

In this paper, we share our experience of designing single-chip multiprocessor controller for advanced multimedia application. We cover the architecture design and validation, the encountered problems and our solutions and we will provide some simulation results.

Keywords: System on Chip, HW/SW Prototyping, ARM, AMBA

1. INTRODUCTION

With rapid advances in semiconductor processing technologies, the density of gates on the die increased considerably. This helped in the realization of more complicated designs on the same IC. This recent progress in the microelectronic enabled the integration of more functionalities in a single chip and it is now possible to create complex embedded system called Multiprocessor System on Chip – MPSoC, containing several microprocessors, memories, shared busies and peripheral circuits - or in other words the whole system on single silicon. This is what has marked the beginning of the SoC era.

We present in this paper the development of a system prototype for validation the software and the hardware parts of a multimedia application. The paper describes the hardware architecture implementation around AMBA bus using IP components integration methodology of a DivX Encoder application.

2. MPEG-4 APPLICATION EXAMPLE

This section presents a multiprocessor video encoder embedded system designed in SLS group at TIMA Laboratory. We developed a DivX Encoder embedded system handling MPEG-4 [1] QCIF resolution (176x144 pixels) at 25 frames/sec to achieve real time video encoding running at 60 MHz.

The DivX Encoder application is based on the OpenDivX algorithm [2]. This algorithm is an open source code implementation of MPEG-4 video compression standard. The compression technique is based on removing spatial and temporal redundancy from input video frames. More details about compression technology, architecture exploration and the software realization of this application could be found in [3] and [4].

2.1. DivX Application Architecture

Figure 1 shows the block diagram of the DivX application architecture obtained after automatic hardware/software refinement using a ROSES [6] environment. It consist of three DivX front-end cores implementing the motion estimation and

compensation, DCT transformation and the quantization, a Variable Length Coder (VLC) back-end core implementing the entropy decoder, a hardware Direct Memory Access engine establishing the communication between all modules and a hardware I/O interface blocks. The four CPU cores are using an ARM946E-S processor.

The functional flow of DivX Encoder application is as follows: The Splitter divides each incoming frame of the input video stream in 3 parts and each frame part is sent to one of the three DivX subsystems: DivX1, DivX2 and DivX3. The DivX cores treat the coming image data and prepare it for compression. The prepared data is transfer to the VLC subsystem, where the compression of the entire frame is finalized and the whole image data is proceed to the Combiner to adjust the compression parameters and transfer it to the system output.

All data transfers between the subsystems are routed through a DMA engine with a point to point connection scheme. This block is particular for the application and it is not a standard Direct Memory Access device adapted for the system purpose. All hardware blocks outside from the subsystems are designed at SLS group, TIMA laboratory in application specific fashion using SystemC [7] language. The software is developed in C/C++ aimed to be executed on ARM processors [8].

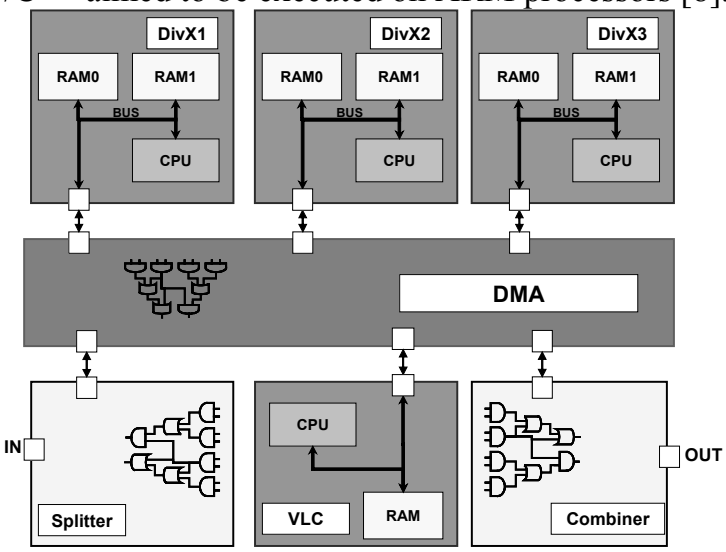


Figure 1: DivX Encoder Application Architecture

3. MAXSIM SIMULATION TOOL

MaxSim [12] is an environment for System-Level Simulation and Design based in SystemC. The designer can construct a virtual prototype by assembling the system from a component library. This component library is composed of processors, buses, memories, interrupt controllers and others. Therefore, the library can be extended with custom components described by the designer.

The computational model of the MaxSim components is based in a cycle-based engine. In this kind of component, the behavior is evaluated only in the clock edges. Behavior is described in two methods, communicate and update. In the communicate method all communication between the components are performed, whereas in the

update method the communication realized are committed in the shared resources. This way of modeling leads to high simulation speeds, enabling the fast validation of the architecture.

MaxSim components can use two kinds of ports: signal or transaction. The signal based connection has the same RTL notion. In the transaction based connection, operations are described by two methods: read and write. A read or write operation is composed of three parameters: address, value and control. These methods always return a flag that indicate if the access was committed or if the resource is not available (WAIT). This is useful for modeling the conflicts or operations that takes mores that one cycle, as for example, a memory access. Also, the control parameter is used to encapsulate the others information specific to the protocol. A component can use these two kinds of interfaces. For example, in the figure 2, we have the ARM9 processor interface composed of signal-level interfaces (fiq, irq, rst), and transaction-level interfaces (mem) to connect the processor with memories modules.

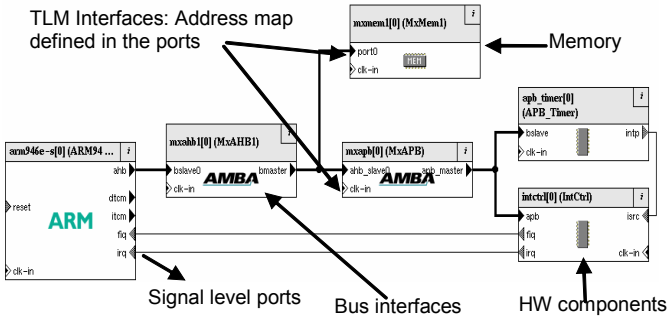


Figure 2: MaxSim representation of one DivX subsystem architecture

The interconnection between components uses explicit structures such as the bus interfaces showed in the Figure 3. This enables the mapping of memories modules and IO components defining the address space for the software development. In addition, MaxSim supports event-based SystemC modules. This can be accomplished by instantiating the SystemC component inside a MaxSim component. An interface adapter is necessary between the SystemC ports and MaxSim ports as showed in the figure 3.

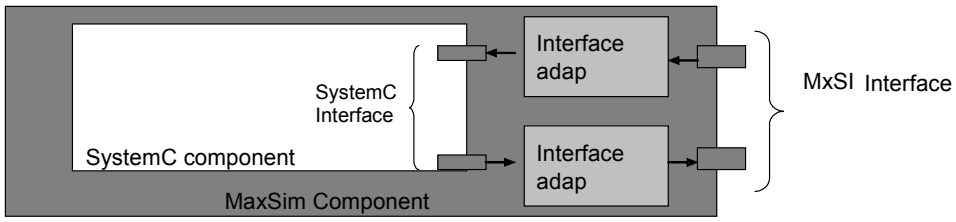


Figure 3: MaxSim SystemC wrapper

For basic SystemC signal-level ports sc_in and sc_out, we implemented generic adapters, facilitating the integration of the SystemC components in MaxSim. This adapter is responsible to convert the SystemC signal-level interface to MaxSim signal-level interfaces. In addition, in MaxSim, the values are exchanged as integer

values, and the adapter makes the type conversion when necessary. This is useful to integrate the RTL models available in SystemC into MaxSim simulations.

4. DIVX ENCODER SUBSYSTEMS

Each DivX Encoder subsystem is based on an ARM946E-S processor and dedicated subsystem architecture around AMBA bus.

The ARM946E-S [8] is a macrocell combining an ARM9E-S processor core with instruction and data caches, tightly coupled instruction and data SRAM memory with protection units, write buffer, and an AMBA AHB interface. The cache architecture is 4-way set associative with data cache support for write-back and write-through policies, and lock down on a per set basis for both instruction and data caches. The ARM9E-S core is a 32-bit RISC processor based on the ARM9TDMI core. It includes new signal processing extensions to the ARM instruction set and a single-cycle 16x32 multiply-accumulate (MAC) unit.

AMBA Bus Architecture. The architecture of the AMBA bus consists of a high-performance bus, called the AHB and a peripheral bus called the APB. The AHB and the APB are connected via a bus bridge. Several slaves are connected to the AHB bus. Some of them are: Default Slave used to respond to transfers addressed to undefined regions of memory, where no AHB slaves are mapped; IRQ Controller providing interface to generate the two ARM9E-S processor interrupt signals from multiple interrupt sources; and AHB-to-APB bridge serving as a slave on the AHB, and the only master in the APB bus.

The various low performance peripherals are attached on the APB bus such as: Timer providing access to two programmable 32-bit free-running decrementing counters; Watchdog providing a way of recovering from software crashes; and Remap/Pause glue logic required to implement correct boot up behavior and a basic low-power mode. All peripheral controllers are parts of AMBA design kit. The main features of this protocol, such as pipelined, burst and split transfers are present in details in AMBA 2.0 specification [8].

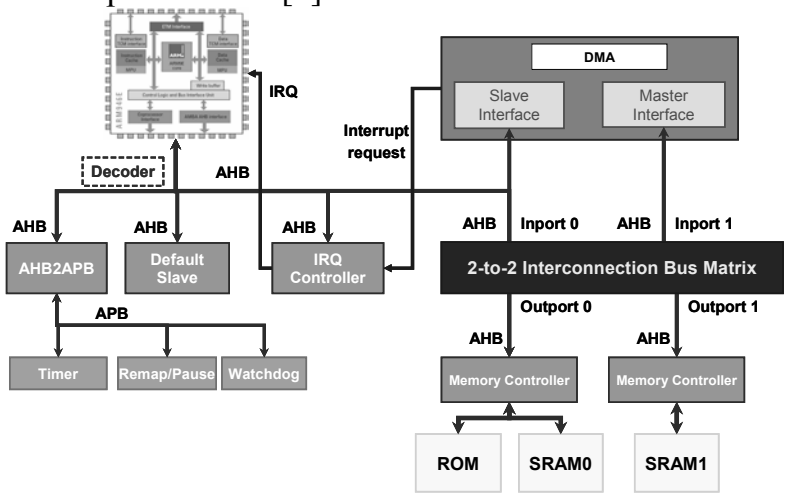


Figure 4: DivX Subsystem Architecture

The DivX core subsystem architecture (see figure 4) has a specific double bank memory support to enable the concurrent data transfer from the DMA and the processor. This functionality is realized with a 2-to-2 interconnection Bus Matrix defined in the specification for Multi-layered AMBA bus [8]. The Bus Matrix allows parallel access to a number of shared AHB slaves from a number of different AHB masters. The switch determines which slave a particular master requires access to and routes the input port to the appropriate slave. Using the Bus Matrix, there is no need to implement an arbiter to allow time distributed access on the AHB bus. Thus, ARM946E-S processor is the only master on the AHB bus and it is all time granted. The Bus Matrix allow to processor and the DMA to work separately in parallel with both bank of memories.

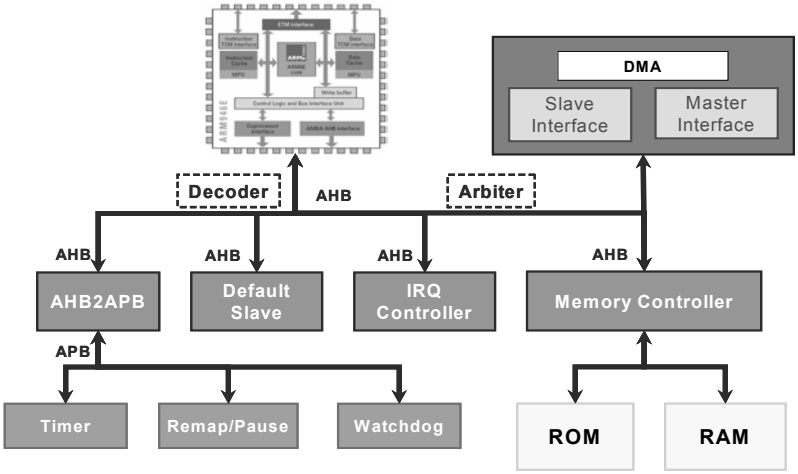
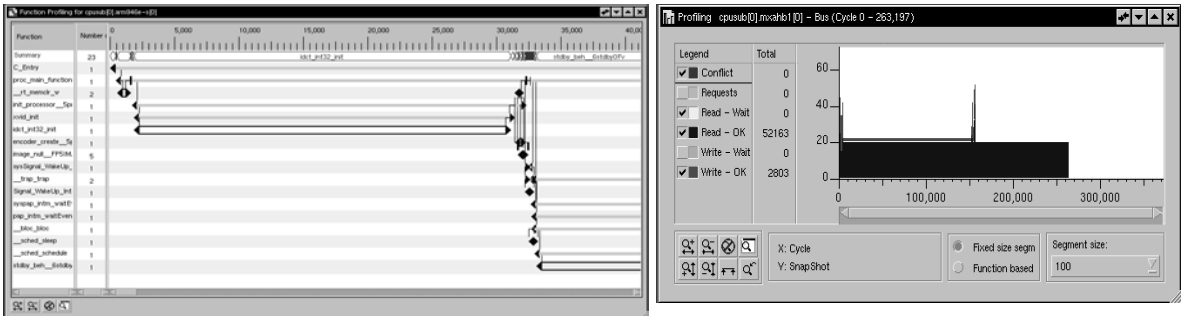


Figure 5: VLC Subsystem Architecture

The VLC core subsystem has the same basic architecture as the DivX core but including only a single memory bank for data buffering (see figure 5). Both subsystems architecture use static synchronous macro cell memories designed for HCMOS9GP technology of ST Microelectronics.

4. EXPERIMENTAL RESULTS



(a) (b)

Figure 6: Software profiling (a) and communication profiling (b) for the encoder subsystem

This section presents the simulation results obtained from the hardware/software simulation of DivX core subsystem.

The virtual prototype was simulated with MaxSim simulation tool to validate the software application in the architecture, and the development of the hardware-dependent software, such as the specific drivers for the interrupt and DMA controller. In addition, the virtual prototype of the system was used for performance analysis of the software and the communication as showed at figure 6.

After the software development and the architecture was validate with the virtual prototype, we build the RTL level model by replacing the functional hardware components their RTL version.

The co-simulation at RTL level was performed on Sun Blade 150 workstation using Cadence NC-SIM [11] for the hardware flow of the system and ARM Development Suite [8] for the software flow. Figure 7 shows the simulation execution of the boot part of the DivX application software on ARM946E-S running at 60 MHz with 4 Kbytes instruction and data caches and 32 Kbytes instruction and data Tightly Coupled Memory.

```
Boot Code
============================================================
   Total ROM    Size (Code + RO Data)              4 320  ( 4.22 Kbytes)
   Total RAM    Size (RW Data + ZI Data)             132  ( 0.13 Kbytes)
============================================================

ARM946E-S Configuration
============================================================
Operating frequency:          60 MHz
Instr CACHE size:          4 096 bytes
Data CACHE size:           4 096 bytes
Instr TCM size:           32 768 bytes
Data TCM size:            32 768 bytes
============================================================

Simulation
============================================================
Execution time: 83286051 PS + 1  ( 83us )
Memory Usage: 19.7M program + 26.5M data = 46.2M total
CPU Usage:       3.6s system + 43.3s user = 46.9s total (229.2s, 20.5%)
```

Figure 7: Simulation results

The verification software includes the initialization and configuration of the processor, the boot sequence, the copying of data from the external ROM to the Instruction Tightly Coupled Memory and execution of the first task of application program. We performed also verification for valid communication transfer between the processor and the DMA.

5. CONCLUSIONS

In this paper, we discuss the development of hardware prototype for ARM based embedded systems to accommodate the requirements for flexibility and testability in the prototyping phase of an embedded system development. Using this design methodology, we were able to validate our architecture for less of one month and to execute a part of the software program to verify for bug errors due the functionality required by the application algorithm.

6. REFERENCES

[1] V. Bhaskaran et al., "Image and Video Compression Standards: Algorithms and Architecture", Kluwer 1995
[2] Open DivX, available at http://www.projectmayo.com/
[3] M.-W. Youssef, S. Yoo, A. Sasongko, Y. Paviot, A.A. Jerraya, "Debugging HW/SW Interface for MPSoC: Video Encoder System Design Case Study", DAC'04, San Diego, USA, June 2004.
[4] M. Bonaciu, A. Bouchhima, M.-W. Youssef, X. Chen, W. Cesasrio, A. Jerraya, "Flexible MPSoC Platform used for Fast Implementation of MPEG4 video Encoder with Custom Parameters", submitted for DAC'05
[5] A. Haverinen, M. Leclercq, N. Weyrich, and D. Wingard. "White Paper for SystemC based SoC Communication Modeling for the OCP Protocol" at www.ocpip.org/data/...
[6] W.O. Cesario, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, L. Gauthier, M. Diaz-Nava, A.A. Jerraya, "Multiprocessor SoC Platforms: A Component-Based Design Approach ", IEEE Design & Test of Computers, Vol. 19 Nr. 6, Nov-Dec, 2002.
[7] SystemC 2.0 available at http://www.systemc.org/
[8] ARM Documentation available at http://www.arm.com
[9] Jason Andrews "ARM SoC Verification Matrix Improves HW/SW Co-Verification" Electronic Design Processes 2004 April 25-27, 2004, Monterey, CA
[10] Russ Klein, Kerry Travilla, Mike Lyons, "Performance Estimation of MPEG4 Algorithms on ARM Architectures Using Co-Verification" Embedded System Conference'04, San Francisco
[11] Cadence available at http://www.cadence.com/
[12] MaxSim available at http://www.arm.com/products/DevTools/MaxSim.html