AUTOMATIC CODE GENERATION FOR MULTI-MICROBLAZE SYSTEM WITH SYNDEX

Pengcheng Mu, Mickaël Raulet, Jean-François Nezan, Jean-Gabriel Cousin

IETR/Image and Remote Sensing Group
CNRS UMR 6164/INSA Rennes
20, avenue des Buttes de Coësmes, 35043 RENNES Cedex, France
pmu@ens.insa-rennes.fr {mraulet, jnezan, jcousin}@insa-rennes.fr

ABSTRACT

Image processing applications such as video codecs represent a great challenge in terms of real-time embedded systems. Programmable multicomponent architectures can provide suitable target solutions combining flexibility and computation power. Integrating multicomponents on FPGA provides greater flexibility but presents more challenges in system level design e.g. design space exploration, multiprocessor distribution and scheduling, inter-processor communications and real-time constraints. The aim of our work is to develop a fast automatic design process dedicated to the implementation of deterministic image processing applications on parallel multicomponent architectures. This design process is based on AAA methodology using the SynDEx CAD tool. A distributed implementation from high-level application and architecture descriptions is automatically provided, saving a considerable amount of time in design space exploration achieving optimisation by reducing global execution time. This paper presents the design process for an FPGA-based multi-MicroBlaze system using SynDEx, and several kernels are developed for the automatic code generation .

1. INTRODUCTION

Today's image processing applications such as video or still image codecs require a lot of processing power. Specific hardware circuits overcome speed constraints but are not compatible with a short time-to-market. They also need early and evaluative demonstration prototypes. An alternative can be provided by programmable software components (such as DSP and RISC processors) or programmable hardware components (such as FPGA components) since they are more flexible. Hard real-time constraints could be satisfied by multicomponent architectures. As DSP is software programmable and FPGA is hardware configurable, this design can provide considerable flexibility. However, this flexibility is limited by the unchangeable topological structure of the programmable software components and their fixed number.

With the help of multi-million gate configurable logic and various heterogeneous FPGA hardware components (multipliers, memory blocks, etc.), soft and hard processors could now be integrated on FPGA. Examples of such soft RISC processors include Nios from Altera[1] and MicroBlaze from Xilinx[2]. In addition, Xilinx has also integrated the PowerPC 405 hard core on their FPGA. In [1], a Texas Instruments[3] C6201 DSP VHDL model is used on FPGA. When the processors are integrated on FPGA, both control and computation functionnalities of the embedded system could be handled by one or more FPGAs, designing an FPGA-based Systems on (Programmable) Chips (SoC). With multiple processors integrated on FPGA, we can also build up Multiprocessor Systems on (Programmable) Chips (MPSoC) [2]. This MPSoC offers flexibility and efficient support, not only as regards its software but also as regards its hardware. Several scenarios for architecture and algorithmic design can be explored to reach real-time constraints. Time-to-market is shorter in comparison with specific hardware circuits but this manual exploration is still complex. As an example of the research for multiprocessor, ATLAS [3], which is also known as RAMP-Red is a prototype including 8 PowerPC 405 cores that run multithreaded code for applications and a ninth core that handles the operation system and I/O devices.

Co-Design [4] is usually used as the design method for embedded systems. When it is used for MPSoC, the multicomponent architecture raises problems in terms of application distribution: manual data transfers and synchronizations quickly become very complex and result in loss of time and potential deadlocks. One suitable design process solution consists of using rapid prototyping methodology. The aim is then to go from a high-level description of the application to its real-time implementation on target architecture as automatically as possible. This automation saves development time and prevents conflicts and deadlocks. It ensures processing safety and reduces validation tests.

A rapid prototyping methodology based on the SynDEx tool is suitable for image processing systems and heterogeneous multicomponent architectures, and has been used in some multi-DSP systems for image processing applications [5]. This paper presents the use of this rapid prototyping methodology in multi-MicroBlaze systems. The document is organized as follows: in section 2 the prototyping methodology is introduced, then it is used for multi-MicroBlaze systems on FPGA in section 3 , and the necessary kernels for automatic code generation are also presented. In section 4, we give the test results and analyze the performances with an application example. Finally, section 5 gives the conclusions and perspectives.

2. AAA FAST PROTOTYPING METHODOLOGY WITH SYNDEX

Our laboratory is studying a rapid prototyping design process based on the use of the SynDEx[4] tool [5]. SynDEx is a free academic system level Computer Aided Design tool

[1]http://www.altera.com
[2]http://www.xilinx.com
[3]http://www.ti.com

[4]http://www.syndex.org

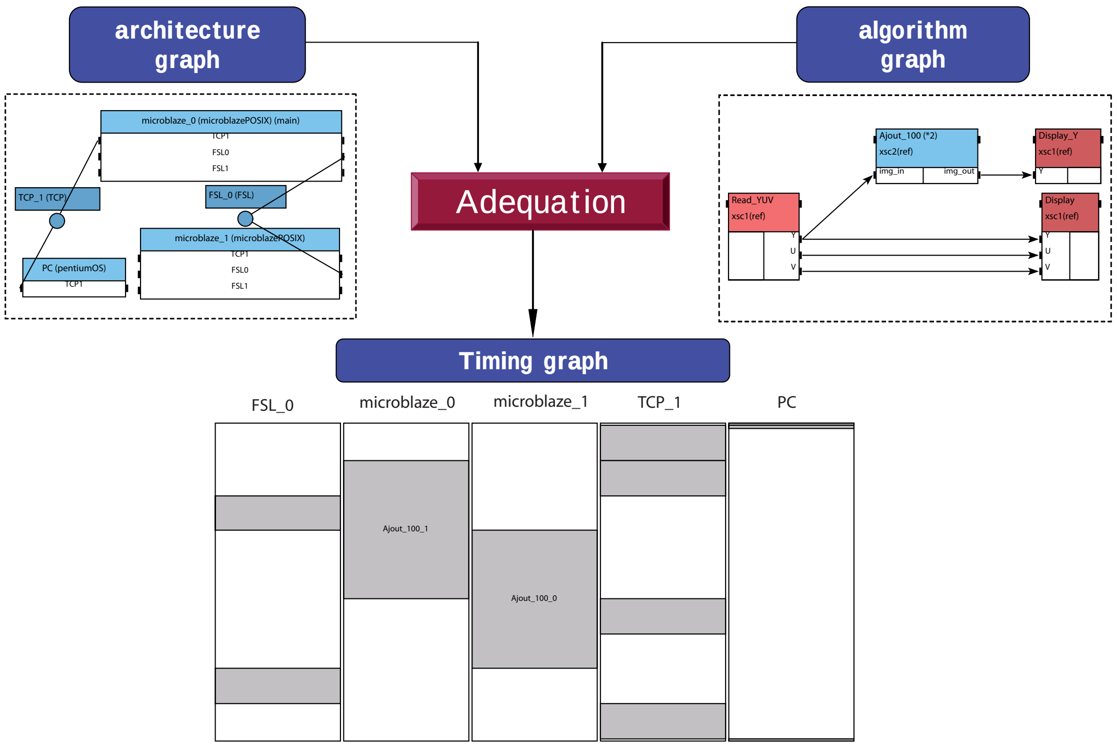

Figure 1: SynDEx design flow

co-developed by INRIA Rocquencourt and our laboratory. It supports the AAA methodology (Adequation Algorithm Architecture [6]) for distributed real-time processing.

2.1 SynDEx tool

The aim of SynDEx is to directly achieve optimized implementation from descriptions of an algorithm and an architecture. Figure 1 gives the SynDEx design flow.

An algorithm graph (Figure 1) is described as a Data Flow Graph (DFG), and specifies the potential parallelism of the application. An architecture graph (Figure 1) describes the multicomponent target, i.e. a set of interconnected processors and specific integrated circuits, and specifies the available parallelism. In the application example given in Figure 1, the algorithm graph includes one input, two outputs and a function which is divided into two identical parts to be executed simultaneously. In the architecture graph, the target is composed of a PC, two MicroBlazes and two communication media that are explained in section 3.5. As these two MicroBlazes and the medium between them are all integrated on an FPGA, the architecture graph gives a medium-coarse grain description in comparison with the one considering an FPGA as a black-box [5].

"Adequation" (Figure 1) means efficient mapping, and consists of manually or automatically exploring the implementation solutions with optimization heuristics [6]. These heuristics aim to minimize the total execution time of the algorithm running on the multicomponent architecture. The heuristic is a greedy list scheduling based approach with manual interaction when timing constraints are not met.

Implementation consists of both performing a distribution (allocating parts of the algorithm to components) and scheduling the algorithm on the architecture i.e. giving a total order for the operations distributed onto a component.

Formal verifications during adequation avoid deadlocks in the communication scheme thanks to semaphores inserted automatically during real-time code generation. Moreover, since the Synchronized Distributed Executives (SynDEx) are automatically generated and safe, part of the tests and low-level manual coding are eliminated, decreasing the development lifecycle.

SynDEx provides a timing graph (Figure 1), which includes simulation results of the distributed application and thus enables SynDEx to be used as a virtual prototyping tool. SynDEx then automatically generates the generic executives, which are independent of the hardware target, and places them in several source files , one for each hardware target.

2.2 Automatic executive generation

The generic executives automatically created by SynDEx are static and composed of a list of macro-calls. The M4[5] macro-processor transforms this list of macro-calls into compilable codes for a specific target. The codes are usually C or assemble codes for processors and VHDL for the specific functions implemented on the FPGA. The M4 macroprocessor replaces macro-calls by their definitions as given in the corresponding executive kernel. The definitions are dependent on a target and/or a communication medium. In this way, SynDEx can be seen as an off-line static operating system that is suitable for setting data-driven scheduling, such as image processing applications. For examples, SynDEx kernels have been developed for several processors such as General Purpose

[5]http://www.gnu.org/software/m4

Processors (usually on PC), Texas Instruments TMS320C6x (C62x, C64x) DSP and Virtex FPGA families [5]. The generated codes could then be compiled by specific CAD tools such as CCS for Texas Instruments DSP, Quartus II or ISE for FPGA and Visual Studio for PC.

2.3 Design process

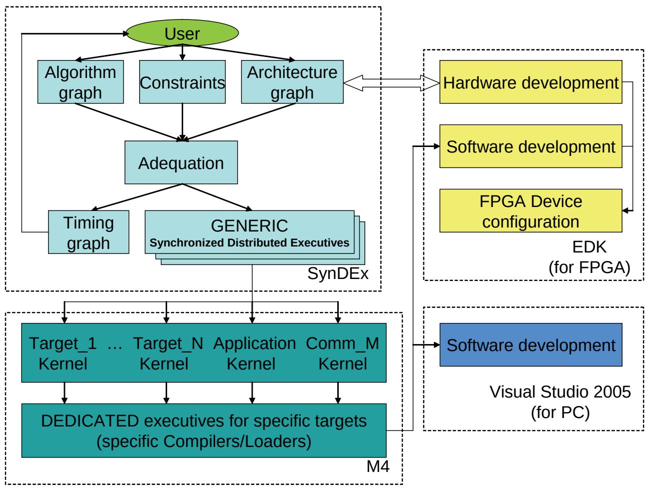

Figure 2: Global prototyping design flow

SynDEx and the use of specific kernels enable full rapid prototyping from the application description (DFG) to final multiprocessor implementation (Figure 2). The DFG description for an application should be done manually while an automatic tool for high-level algorithmic complexity analysis may be used to help the system design [7].

The main advantage of the SynDEx based prototyping process is its simplicity because most of the tasks performed by the user concern the description of an application (creation of the algorithm graph) and a compiling environment. All complex tasks (adequation, synchronization, data transfers and chronometric reports) are executed automatically or semi-automatically. The user can rapidly explore several design alternatives by modifying the architecture graph and/or the algorithm graph, or by adding constraints [5].

3. FAST PROTOTYPING FOR MULTI-MICROBLAZE SYSTEMS ON FPGA WITH SYNDEX

The AAA fast prototyping methodology has been used in a number of multi-DSP systems for image processing applications, and it could also be used in FPGA-based MPSoC to integrate multiple components on one or more FPGAs. As an embedded soft core, MicroBlaze is a RISC processor and optimized for implementation in Xilinx FPGA. It is highly configurable, allowing users to select a specific set of features required by their design. Integrating multiple MicroBlazes on one or more FPGAs can build up a multi-MicroBlaze MPSoC. This multi-MicroBlaze system is flexible in terms of both software and hardware, so it can be used in complicated and computation-rich applications such as image processing. This section details the use of fast prototyping for multi-MicroBlaze systems on FPGA with SynDEx.

3.1 Prototyping and design flow

Figure 2 shows the design flow for multi-MicroBlaze systems, a number of tools such as SynDEx, M4, EDK and Visual Studio are necessary for the design stage.

In this design flow, users firstly have to model in SynDEx IDE (Integrated development environment) the processors and communication media which are used in their design. Two different models are possible for the communication media between processors: SAM (Single Access Memory) and RAM (Random Access Memory, shared memory) [5].These modules are saved in a library and could be used for other designs without any modifications. With these modules, users then could build up the architecture and the algorithm graphs, and the adequation would be done by SynDEx while the Synchronized Distributed Executives are automatically generated in the form of m4 files. The m4 files then are translated into compilable executives for specific targets such as MicroBlaze, PC and specific functions on FPGA with the help of kernels which are explained in this section.

As the codes are generated, the next step is to build up the system. For the Xilinx FPGA, EDK (Embedded Development Kit) is used for both hardware and software developments. The hardware is equivalent to the description of FPGA in the architecture graph of SynDEx except that it is described in the finest grain with EDK and can be used to generate the bitstream that configures the FPGA. For software programming, EDK uses the generated executives for MicroBlazes and respective drivers to build up ELF files (Executable Linked Format) for the multi-MicroBlaze system. When PC is used, Visual Studio (or other tools such as Dev-C++) is necessary for software development using the generated executives for PC and respective drivers.

3.2 Development platform for multi-MicroBlaze systems

The ML402 development board and its complete development line have been used in our work. This board is designed for applications in Digital Video, DSP, Image Processing. Its main features are as follows:

- Xilinx Virtex4 FPGA with 100MHz Oscillator for system clock
- Different types of memories including DDR SDRAM, ZBT SRAM, Flash and IIC EEPROM
- A set of connectors and interfaces including 10/100/1000 Tri-Speed Ethernet PHY, USB controller with Host and Peripheral Ports, DB 15 VGA display, JTAG Configuration Port etc.

In addition to the board, a PC has been used to act as the first producer and last consumer of data. It connects to the board via an Ethernet network, which allows TCP/IP communication between PC and the board (MicroBlaze in our application). The most important advantage of such a platform is that multiple processors and coprocessors can be integrated on the FPGA, and their topological structure could be customized to satisfy a user's needs. In fact, the structure usually depends on the application. Multiple types of medium such as FIFO, shared memory can be used for communication between processors and an example of a system composed of three MicroBlazes and a PC is showed in Figure 3 where the three MicroBlazes are integrated on the FPGA of the board.

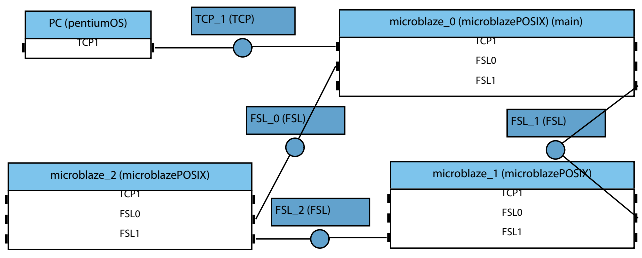

Figure 3: Architecture of a multi-MicroBlaze system

3.3 SynDEx executive kernels

As described in section 2, the SynDEx generic executives have to be translated into a compilable language. The translation of SynDEx macros into the target language is contained in library files (also called kernels). Figure 4 shows the organization of different kernels for the multi-MicroBlaze system as detailed in the following sections.

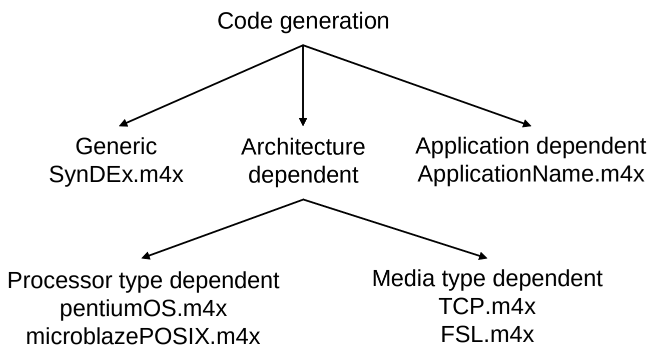

Figure 4: SynDEx kernel organization

3.4 MicroBlaze kernel

The program in MicroBlaze-based systems could be designed using either a standalone Board Support Package (BSP), which has no operating system, or Xilkernel, which supports the core features required in an embedded, real-time operating system (RTOS). Xilkernel is a POSIX compliant API. When using Xilkernel, the standalone BSP is used below the operating system layer. In [8], RTOS is introduced in the AAA methodology. The RTOS has an impact on processor target such as execution time or allocated memory, but the overcost is slight, especially for image processing algorithms where data in the DFG are often large. Moreover, executives automatically generated including RTOS primitives are simple leading to a better comprehension for users. They are also more generic and compatible with more components. Therefore, our software component kernel for MicroBlaze has been developed using Xilkernel.

A software component kernel is used to automatically generate executives that would run in a specific processor, and different kernels should be used for different processors. With the MicroBlaze kernel, the generic executives generated by SynDEx are translated into MicroBlaze compilable C codes. The generated codes are compiled using Xilkernel, and semaphores are used to synchronize the various threads of the program.

Executives generated by SynDEx consist of a sequential list of function calls (one for each DFG operation). Therefore, functions have to be defined outside of SynDEx to make the whole program executable. Most of these functions are developed in C language so that they can be reused for any C programmable device.

3.5 Communication media kernels for multi-MicroBlaze system

As described in 3.1, communication media are important components in the fast prototyping methodology. Two types of communication media are used in our multi-MicroBlaze system, one for the communication between MicroBlazes and the other for the communication between MicroBlaze and PC. In our work, Fast Simplex Link (FSL) is used for the former and TCP/IP for the latter.

3.5.1 FSL

FSL is a uni-directional point-to-point communication channel bus used to provide fast communication between two IPs. Up to 8 master and slave FSL interfaces are available on a MicroBlaze processor, and the number of FSL can be customized according to design needs. As FSL is a FIFO-based communication bus, a SAM is used in SynDEx to model it and a kernel has been developed for it.

FSL can be used to transfer data in two clock cycles between registers on the processor and hardware running on the FPGA. The hardware can be either an IP peripheral (MicroBlaze coprocessor) or another MicroBlaze processor (multi-MicroBlaze).

3.5.2 TCP/IP

TCP/IP is a widely used communication protocol. The Ethernet hardware is provided in the ML402 development board as well as an IP to control it: the Ethernet Media Access Controller (EMAC). The EMAC and a MicroBlaze are both on the FPGA and internally connected by an OPB bus (On-chip Peripheral Bus).

For image processing applications, TCP/IP could be used to transfer data from PC to ML402 board for processing, as well as transferring data to display the results (processed images) on PC. In SynDEx, TCP/IP could be modeled as a SAM because it uses FIFOs. With the kernel developed for TCP/IP, SynDEx could generate a sequence of generic executives to complete TCP/IP-based communication. Like the computation function requirements, the communication functions should also be developed outside SynDEx, and these functions may be different depending on the different types of processors. C functions have been respectively developed for PC and MicroBlaze so that they can communicate using TCP/IP.

To program TCP/IP on a PC, Windows Sockets 2 is commonly used, while for MicroBlaze, two libraries could be used, according to the user's requirements. One of the network libraries for embedded processors is libXilNet, which is developed by Xilinx and provides a simple set of Socket Application Programming Interface (APIs) functions; the other is the Light Weight IP (lwIP) library, which is a third party network library supporting two interfaces - Raw API and BSD style Socket API. For comparison, libXilNet consumes less memory space and has more restrictions while the lwIP library is much more powerful but needs more memory.

Users can choose one of these two libraries according to their requirements e.g. throughput for communication, memory capacity etc. In our application, libXilNet is used for reasons of simplicity and we can achieve a throughput of 10Mbps.

4. RESULTS

A simple image processing application is shown in Figure 1. Using our kernel for TCP/IP, a throughput of 10 Mbps has been achieved. Since we have used libXilNet as the network program library for MicroBlaze, which supports only the polled mode, no significant improvement could be obtained in throughput in this case. However, EMAC also supports other two types of MAC modes such as simple FIFO interrupt mode and DMA (Direct Memory Access) mode (including simple DMA and Scatter/Gather DMA). To make EMAC work under these modes, the lwIP library should be used in place of libXilNet. In fact, lwIP supports both the polled mode and the simple FIFO interrupt mode with Raw API as well as simple FIFO interrupt mode and simple DMA mode with Socket API; Scatter/Gather DMA mode is not currently supported by lwIP. Using lwIP could improve throughput levels but more memory should be provided for the program. This means that a tradeoff is required between throughput and memory.

As the FSL interface is very simple, FSL does not support interrupt; furthermore, as FSL is used to transfer data between the dedicated register on MicroBlaze and other hardware running on the FPGA, it does not support DMA. So FSL-based data transfer should always be controlled by the processor. As a result, FSL is suitable for connecting the MicroBlaze and the IP coprocessor because the data could be transferred to the coprocessor and be retrieved from the coprocessor after processing without excessive delays. When FSL is used to send data from one MicroBlaze to another, as the destination MicroBlaze is usually processing multiple tasks, the source MicroBlaze may have to wait until the destination MicroBlaze is free to receive the data. This being so, FSL is a simple means of communication between two MicroBlazes but it has its drawbacks.

5. CONCLUSIONS AND PERSPECTIVES

This paper has presented the AAA fast prototyping methodology used on the multi-MicroBlaze system with the SynDEx tool and a given design process. One software component kernel for MicroBlaze is developed for automatic code generation, and the generated code supports Xilkernel RTOS. Different kinds of communication kernels have been developed such as TCP/IP for communication between PC and MicroBlaze as well as FSL for communication between MicroBlazes. Using this design process, the most important and complicated parts of multi-processor development, such as the distribution of code for different processors, or synchronization between computation and communication are all implemented by the SynDEx tool, and the compilable C code could be generated automatically with the help of the necessary kernels. As the FPGA-based multi-MicroBlaze system can have a very flexible architecture that may change greatly depending on the application algorithm, the use of this method can facilitate development and save a considerable amount of time, design space and the corresponding application optimization.

Though FSL is a simple FIFO-based communication channel between MicroBlazes, it does not support DMA and interrupt. A new type of FIFO should therefore be developed in the future, supporting DMA to allow parallel computation and communication. Some other types of communication based on shared memory would also be attractive and could be developed as a RAM model in SynDEx for future use.

As the complexity of the application increases, more powerful computation ability will be needed and IP coprocessors are always used for time-consuming computations. Because of simultaneous execution, the time required for computation could be greatly reduced and this is very important for real-time embedded systems. As MicroBlaze is a soft processor integrated on FPGA, connecting a user IP to MicroBlaze would be very convenient and flexible. Therefore, developing hardware component kernels for IP coprocessors in SynDEx would be necessary for the AAA fast prototyping of multi-MicroBlaze systems when the application becomes complicated and time-consuming.

REFERENCES

[1] Vincent Brost, Fan Yang, and Michel Paindavoine, "Rapid implementation of image processing onto fpga using modular dsp c6201 vhdl model," in *The 27th International Congress on High-Speed Photography and Photonics*, Xian, China, 2006.

[2] Grant Martin, "Overview of the mpsoc design challenge," in *Proceedings of the 43rd annual conference on Design automation*, San Francisco, CA, USA, July 2006.

[3] Njuguna Njoroge, Jared Casper, Sewook Wee, Yuriy Teslyar, Daxia Ge, Christos Kozyrakis, and Kunle Olukotun, "Atlas: A chip-multiprocessor with transactional memory support," in *Proceedings of the Conference on Design Automation and Test in Europe (DATE)*, Nice, France, April 2007.

[4] G. De Michell and R. K. Gupta, "Hardware/software co-design," *Proceeding of the IEEE*, vol. 85, no. 3, pp. 349–365, March 1997.

[5] M. Raulet, F. Urban, J.-F. Nezan, C. Moy, O. Déforges, and Y. Sorel, "Rapid Prototyping For Heterogeneous Multicomponent Systems: An MPEG-4 Stream Over An UMTS Communication Link," *Journal Of Applied Signal Processing (JASP)*, 2005.

[6] T. Grandpierre, C. Lavarenne, and Y. Sorel, "Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors," in *Proceedings of 7th International Workshop on Hardware/Software Co-Design, CODES'99*, Rome, Italy, May 1999.

[7] Massimo Ravasi and Marco Mattavelli, "High abstraction level complexity analysis and memory architecture simulations for multimedia algorithms," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 15, no. 5, pp. 673–684, May 2005.

[8] Ghislain Roquier, Michaël Raulet, Jean-François Nezan, and Olivier Déforges, "Using RTOS in the AAA methodology automatic executive generation," in *Proceedings of 14th European Signal Processing Conference*, Florence, Italy, 2006.