IMPLEMENTING REAL-TIME ALGORITHMS BY USING THE AAA PROTOTYPING METHODOLOGY

Pierre Niang, Thierry Grandpierre, Mohamed Akil
ESIEE Paris, Lab. A^{2}SI, Cite Descarte BP 99, Noisy Le Grand 93160, Cedex France
niangp@esiee.fr; t.grandpierre@esiee.fr; akilm@esiee.fr

Abstract This paper presents a system-level methodology (AAA) for signal and image processing algorithms onto circuit architecture. This AAA (Algorithm Architecture Adequation) methodology is added and implemented in an existing software dedicated to the fast prototyping and optimization of real-time embedded algorithms onto multicomponent architectures. The resulting tool, called SynDEx-IC, is a free graphical Computer-Aided Design (CAD) software. It supports different features: algorithm and architecture specifications, data path and control path synthesis, performances prediction, optimizations and RTL generation.

Introduction

Digital signal processing applications, including image processing algorithms, require growing computational power especially when they are executed under real-time constraints. This power may be achieved by the high performance of multicomponent architectures based on programmable components (processors) which offer flexibility and non programmable components (reconfigurable circuits) which offer higher performances with less flexibilities. Consequently, there is a need for dedicated high level design methodology, associated to efficient software environments to help the real-time application designer to solve the specification, validation optimization and synthesis problems. Several research efforts have addressed the issue of design space exploration and performance analysis of the embedded systems. Therefore, some methodologies and tools have been developed to help designers for the implementation process. Among these different researches for multicomponent designs, one may cite the SPADE methodology [LWVD01] and the CODEF tool [ACCG01]. Although these tools are very efficient, none of them is able to bring together under real-time and resources constraints: unified models, graphical specification, performances prediction, generation of distributed and optimized executives for programmable part and generation of optimized RTL

code for configurable part. Actually, the tool associated to AAA methodology [GS03] is named SynDEx. It supports mainly multiprocessor architectures but does not allow to address the optimization and VHDL generation process of configurable part. Hence, there is a need to extend the AAA for circuit and to develop an associated tool. The presented work is an intermediate step for tending towards a codesign tool associated to the AAA methodology. There are several tools allowing to automate the optimized hardware implementation for reconfigurable circuits from a high-level design specification. One may cite Esterel Studio tool, where the user captures a design specification and then automatically generates the hardware description in HDL-RTL. Another high-level synthesis framework, SPARK [GDGN03], provides a number of code transformations techniques. SPARK takes behavioral C code as input and generates Register Transfer Logic (RTL) code. This RTL code may be synthesized and mapped onto an FPGA. One may also cite Simulink which is an extension of MATLAB that allows to create algorithm in a graphical fashion. This allows the engineer to visually follow algorithms without getting lost in low level code. Simulink uses a System Generator which takes this graphical algorithmic approach and extends it to FPGA development by using special Simulink blocks. However, none of them is integrated or interfaced with a codesign tool for the multicomponent implementations. The remainder of this paper is centered on the AAA/SynDEx-IC and the implementation of image processing applications by using SynDEx-IC tool. In Section 1, the transformation flow used by AAA methodology is introduced. In Section 2 the software tool SynDEx-IC which implements the AAA methodology for circuits is presented. Section 3 introduces the implementation of image processing onto an FPGA. Finally, Section 4 concludes and discusses the future work.

1. AAA Methodology for integrated circuits

AAA is supported by SynDEx tool which is based on dedicated heuristics for the distribution and scheduling of a given algorithm onto programmable components. SynDEx uses graph theory in order to model multiprocessor architectures, applicative algorithms, the optimization and code generation. We will extend the AAA methodology for integrated circuits. In the case where the implementation does not satisfy the constraints specified by the user, we apply an optimization process in order to reduce the latency by increasing the number of circuit resources used.

Algorithm specification

The algorithm is modelled by an oriented hyper-graph G_{al} of operations (graph vertices O), its execution is partially ordered by their data-dependences (oriented graph edges D with $D, G_{al} = (O, D)$). In the algorithm graph, each

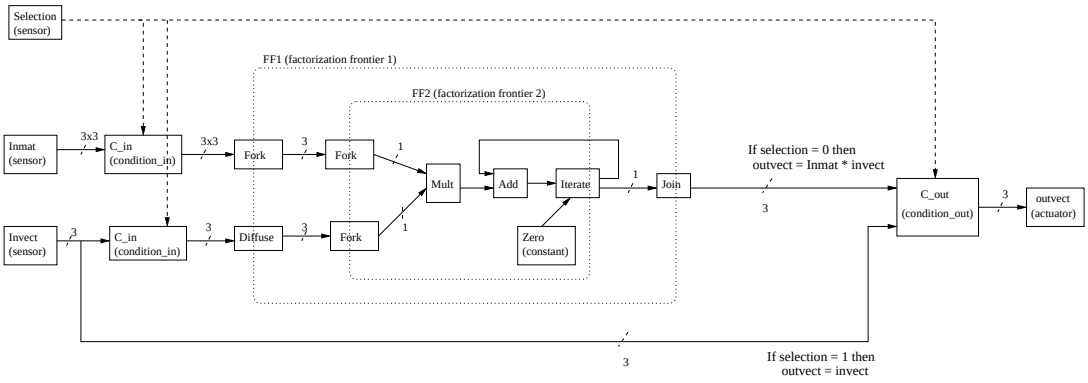

Figure 1. Algorithm graph of CMVP

vertex represents a computation operation, an input/output operation, a conditioned operation (if then else) or a finite factorization frontier for repetitive subgraphs (for i=x to y): a finite factorization frontier is an abstraction delimited by factorization frontier vertices (Fork, Diffuse, Join, Iterate), an infinite factorization frontier is the interface with the external environment delimited by input/output operation. This data-dependence graph, also called directed acyclic graph (DAG), exhibits a potential parallelism. It is important to note that the factorization serves for simplifying the algorithm specification and it is used to explore different optimized implementations. It does not involve a sequential execution inevitably (i.e a "loop") in the circuit. To illustrate the algorithm model, the figure 1 presents a Conditioning Matrix-Vector Product (CMVP). If "selection" input is equal to 1 then "outvect" output copy the "invect" input else if "selection" input is equal to 0 then we have the Matrix-Vector ("inmat" and "invect") product on the "outvect" output. The choice of this example was motivated by regular computations on different array data which highlight the use of the factorization process.

High level synthesis based upon graph transformations

The hardware implementation of the factorized data dependence graph consists of providing a corresponding implementation for the data path and the control path of the algorithm. In this part of the paper, the high level synthesis of the control path is introduced [LAGS04]. The control path corresponds to the logic functions to add to the data path, in order to control the multiplexers and the transitions of the registers composing the modules performing the graph operations. The control path is then obtained for synchronization of data

transfer between registers. In order to synthesis the control path, SynDEx-IC starts by building the neighborhood graph corresponding to the algorithm graph. The neighborhood graph is an interconnection graph of the factorization frontiers, the sequential operations (operation driven by a clock), the operations specific to conditioning (Condition_In, Condition_Out) and the combinative operations (operation whose response at the moment t depends only on its input at the moment t). According to the data dependences relating theses vertices, every vertex may be a consumer or/and producer relatively to another vertex. The control path graph is composed of CU vertices (Control Unit) connected between them according to the connections relations of the neighborhood graph. Moreover, if the consumer data comes from various producers with different propagation time, it is necessary to use a synchronized data transfer process. The synchronization is possible through a request/acknowledge communication protocol. Thus, the synchronization of the circuit implementing the algorithm is reduced to the synchronization of the request/acknowledge signals of the set of CU vertices. In the control path graph, we have four kinds of CU vertices: CU of factorization frontier, CU of a sequential operation, CU of a combinative operation block delay and CU of a conditioning operation.

Optimization of the implementation

The target architecture is made of one reconfigurable circuit (FPGA). Once the algorithm and architecture are specified, it is necessary to characterize each operation in order to be able to perform the implementation optimization process. In this context, the designers have to specify the latency time and the logical units number occupied on the FPGA by each operation of the algorithm graph. The information can be obtained by traditional synthesis tools. A heuristic was developed and implemented to check an adequate implemen-

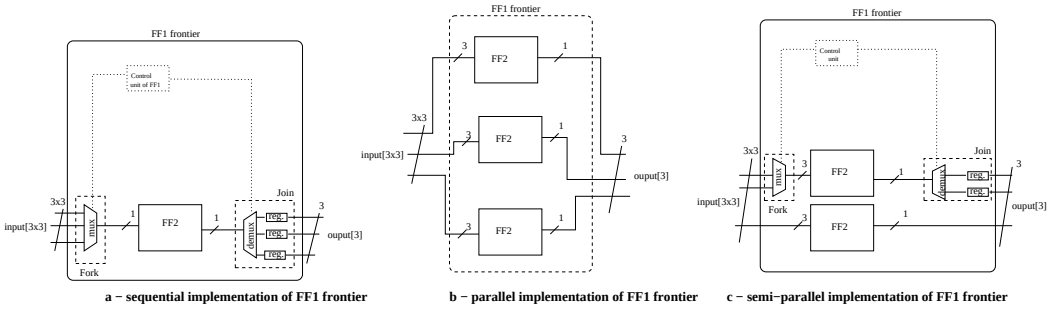

Figure 2. Three examples of implementation of FF1 frontier of the CMVP

tation for the available circuit while satisfying the constraint latency which be specified by designers. For each repeated hierarchical operation, there are several possible implementations. On an FPGA, the CMVP can be implemented either in a purely sequential way (figure 2-a), either in a purely parallel way (figure 2-b) or in a semi-parallel way i.e. mixing the sequential and parallel implementation (figure 2-c). The sequential implementations require the synthesis of a dedicated control path (constituted of multiplexers, demultiplexers, registers and control units). The purely sequential implementation (a) uses less surface (number of logical units occupied by the set of algorithm graph operations) but is slower. The purely parallel implementation (b) is faster but occupies the maximum of surface on the FPGA. The implementation semi-parallel (c) is an example of trade-off between the occupied surface and the latency time, it generates an execution time twice faster than that of (a) and requires also about twice more surface. The combination of all the possible implementations of the repeated operations constitutes the implementations space which have to be explored in order to find a optimized solution. Consequently, for a given algorithm graph, there is a great number of possible implementations. Among all these implementations, we have to choose one satisfying the real-time latency constraint and fitting into the FPGA. This is why we developed heuristic ones guided by a cost function. The cost function allows to compare only a part of the solutions which seems most interesting of this implementations space. Thus, the heuristic is based on an greedy algorithm rapid and effective whose cost function f is based over the critical path length of the implementation graph: it takes into account the latency of the application T and the implementation surface A which are estimated (performance prediction) by SynDEx-IC starting from the characterization.

2. SynDEx-IC: a software framework

SynDEx-IC tool is developed starting from the version 6.6.1 of SynDEx. The coding was performed in CAML language as for SynDEx. The design flow of SynDEx-IC presented in figure 3 is made up of three parts which will be introduced below.

Applications specification

This part is the starting point of SynDEx-IC design flow (Cf part A of the figure 3). The figure 4 details the graphical interface for the algorithm specification: each box represents either a computation operation ("Add", "Mul") or an input/output operation ("input", "output") which is used to transmit data from one iteration to another. Then, it is necessary to specify: the algorithm latency constraint (execution time of all the application) and the features of the target FPGA (for each element of the algorithm graph, it is necessary to

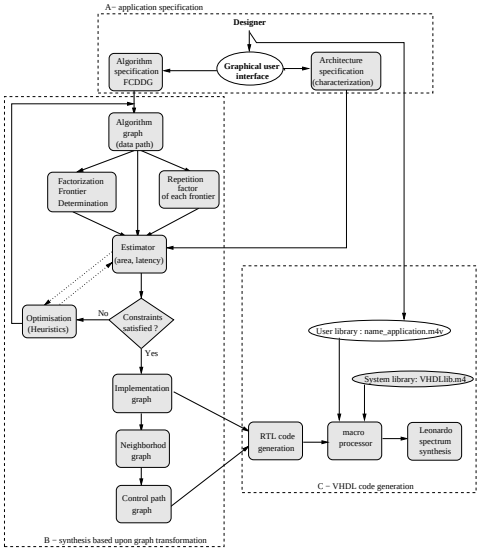

Figure 3. The AAA/SynDEx-IC design flow

define its latency and the quantity of resources necessary to its material implementation on the target component). Besides, the user needs to specify the performing frequency of all the sequential operations used in the algorithm.

Graph transformation and optimization

This section is the core of SynDEx-IC design flow (Cf part B of the figure 3). From the algorithm graph (data path) of the application, the graph transformation of all the graph is proceeded. The function of graph transformation allows the insertion of the specific operations which make it possible to mark the repetition and of the conditioning operations: Fork, Join, Iterate, Diffuse, Condition_In, Condition_Out. Thus, the determination of the frontiers and of the factorization ratio of each frontier is processed before to begin the optimization heuristic. It seeks an optimized implementation graph of the application which satisfies the latency constraint. This implementation graph is used to build the neighborhood graph in order to synthetize the control path.

Automatic VHDL code generation

In order to implement the application algorithm onto the corresponding circuit we need to generate the data path as well as the control path. This code generation is done by way of an intermediate macro code which is used to have a code generator independent of the hardware description language Each op-

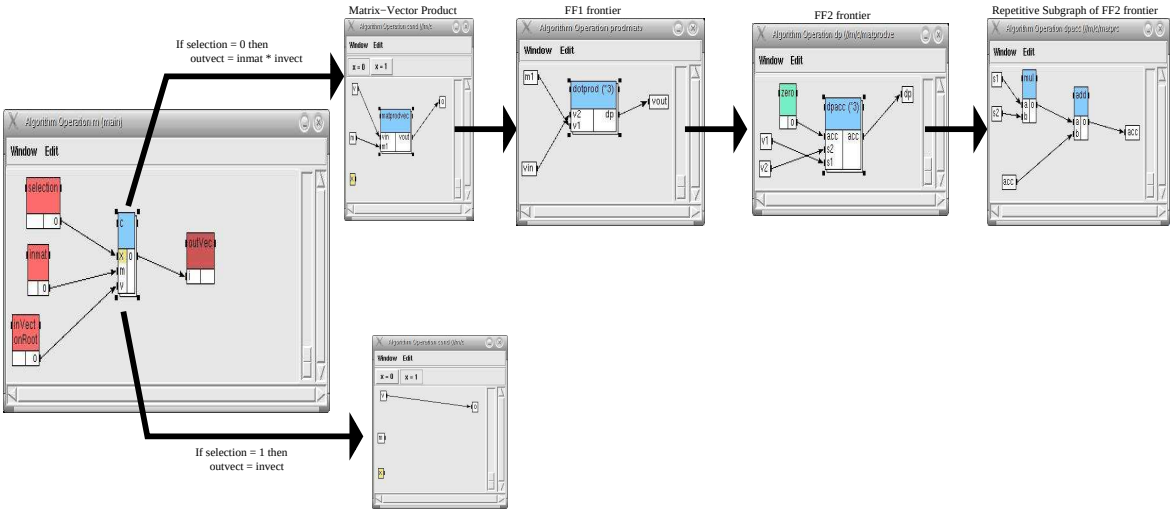

Figure 4. SynDEx-IC snapshot of a CMVP

eration of the optimized implementation graph corresponds to a RTL module which will be implemented on a reconfigurable component.

3. FPGA implementation of the processing algorithms

In order to illustrate the use of SynDEx-IC tool, real time algorithms in image processing were implemented onto Xilinx FPGA (spartan XC2S100). The aim of the work is to carry out an application of video processing using an electronic card based on FPGA. Video processing algorithms were implemented as well as an electronic card for interfacing the FPGA with a video camera and a monitor. The FPGA treating only digital signals, it is necessary to digitize the composite video signal. Thus, one will use an analogical/digital converter (ADC) so that the FPGA can process the pixels, as well as a digital/analogical converter (DAC) to restore an analogical signal at exit. Moreover to process sequence a video, it is necessary to use the synchronization signals of the video. For that, one will use an extractor of synchronization.

Implementation onto FPGA using SynDEx-IC software

Several filters (robert, prewitt, sobel, canny-deriche, low pass) were developed. However, in this paper only sobel and prewitt filters (contours extraction) will be introduced. In order to implement these filters, one started by specifying the data-flow graph of the application, the latency constraint and

then the characterization of the target component. Thus, to obtain optimized VHDL code of the application, SynDEx-IC performs the heuristic of optimization. The algorithmic specification of filters may be obtained from the transform in Z of the transfer function. This specification can also be made starting from the application of a convolution matrix. The transfer function of the so-

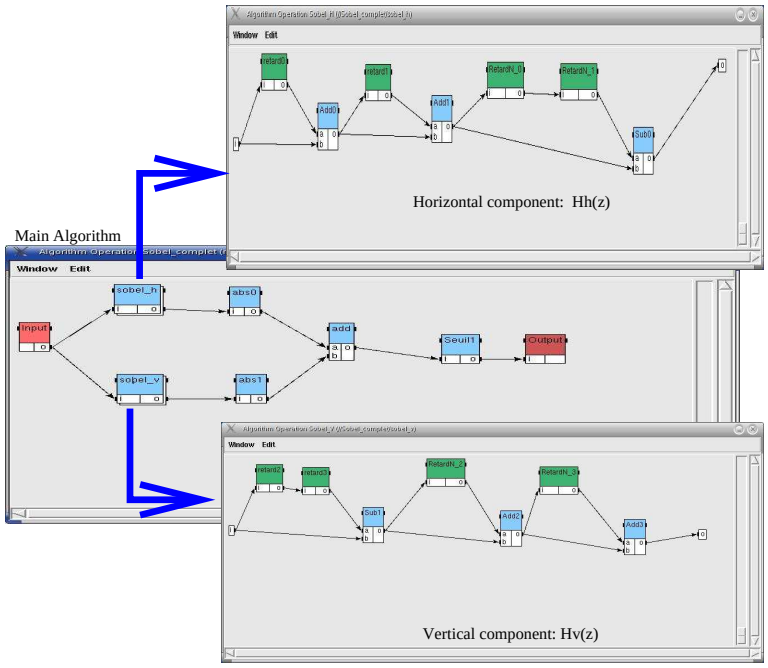

Figure 5. snapshot of sobel filter

bel filter is divided in two parts: the horizontal component and the vertical component. The transfer function of the horizontal component is presented as follows: $Hh(z) = (1 + z^{-1})^2(1 - z^{-2N})$
while the vertical component: $Hv(z) = (1 - z^{-2})(1 + z^{-N})^2$
The figure 5 is a SynDEx-IC specification of the sobel filter using the transfer function. In this specification, the initial image is treated by each component (horizontal and vertical) of the filters, then the absolute values ("abs") of the pixels of the two resulting images are summoned two to two (the calculation approximation of the gradient result amplitude: modulate of the two images resulting). Finally, we have a function of threshold ("seuil") which makes it possible to lower the level of gray of the image result. The sensor "input" represents the input pixels while the actuator "output" represents the output pixels. This algorithmic specification needs also two storage operations: a component

Filters	Number of CLB	Number of Flip Flop	Number of Ram block	critical time (ns)
Sobel (transfer function)	376 of 1200	397 of 2400	8 of 10	36.066
Sobel (hand filter)	334 of 1200	356 of 2400	8 of 10	31.174
Prewitt (transfer function)	579 of 1200	424 of 2400	10 of 10	35.946
Prewitt (hand filter)	525 of 1200	386 of 2400	10 of 10	29.572

Table 1. comparison table of the filters synthesis

"retard" allowing to store a pixel during a signal of clock and a component "retardN" allowing to store a pixel throughout all a line.

Implementation results

Starting from code generated by SynDEx-IC, one simulated the filters of sobel and prewitt by using the modelsim tool. Thus, one obtained images results enough interesting for various input images. If one compares the results obtained for code generated by SynDEx-IC with the results obtained by coding these filters in language C, one noted that they were identical. Thus, for somebody who knows neither the VHDL, nor the language C it may be more practical to use graphical specification SynDEx-IC to design these filters of image processing. Once the simulation of generated code made, one proceeded to the implementing of these filters on the Spartan by using the xilinx webpack tool. These logic synthesis results seen in the TABLE 1 show that compared to the hand filters (make by a manual designer), the SynDEx-IC filters are less fast but easier and faster to design for the user. A reduction area estimated to about 10% and a increasing delay time estimated to about 15% are achieved for hand compared to SynDEx-IC.

4. Conclusion and Outlook

Given a single data-flow graph specification, it is possible to generate a multiprocessor optimized implementation using SynDEx or an FPGA optimized implementation using SynDEx-IC. Moreover, the development flow is unified from the application designer point of view (figure 6). We have presented the different steps to generate a complete RTL design corresponding to the optimized implementation of an application specified on SynDEx-IC.

This work is an intermediate step in order to finally provide a methodology allowing to automate the optimized hardware/software implementation. The next step will be to merge SynDEx-IC and SynDEx in order to support **mixed** parallel architectures (architectures with programmable components **and** reconfigurable circuit). To support mixed parallel architectures, the partitioning problem between programmable and configurable components should be

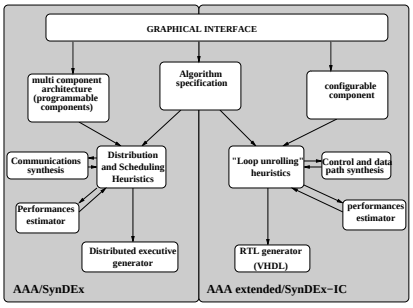

Figure 6. AAA methodology and its extension

resolved in first. For that purpose, it will be necessary to connect SynDEx and SynDEx-IC heuristics. We are developing new hardware/software partitioning heuristics for programmable and configurable components. Thus, the automatic communication synthesis between these two parts is being studied. This work is supported by the Conseil Régional d'Ile de France (PRIMPROC Project).

References

[ACCG01] M. Auguin, L. Capella, F. Cuesta, and E. Gresset. Codef: A system level design space exploration tool. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, page 4, Salt Lake City, USA, may 2001.

[GDGN03] S. Gupta, N. Dutt, R. Gupta, and A. Nicolau. Spark, high-level synthesis framework for applying parallelizing compiler transformations. In *Intl. Conf. on VLSI Design*, Mumbai, India, January 2003.

[GS03] T. Grandpierre and Y. Sorel. From algorithm and architecture specifications to automatic generation of distributed real-time executives: a seamless flow of graphs transformations. In *First ACM & IEEE Intl. Conf. on formal methods and models for codesign. MEMOCODE'03*, Mt Saint-Michel, France, june 2003.

[LAGS04] L.Kaouane, M. Akil, T. Grandpierre, and Y. Sorel. A methodology to implement real-time applications on reconfigurable circuits. In *Special issue on Engin. of Config. Systems of the Journal of Supercomputing*. Kluwer Academic, 2004.

[LWVD01] P. Lieverse, P. Van Der Wolf, K. Vissers, and E. Deprettere. A methodology for architecture exploration of heterogeneous signal processing systems. In Journal of VLSI Signal Processing Systems, editor, *Special issue on signal processing systems design and implementation*, volume 29, page 10, Hingham, MA, USA, November 2001. Kluwer Academic Publishers.