Modular VLIW processor based on FPGA for real-time image processing

VINCENT BROST, DEBYO SAPTONO, CHARLES MEUNIER, FAN YANG, DOMINIQUE GINHAC

LE2I-CNRS 5158 Laboratory, University of Burgundy, 21078 Dijon France

vincent.brost@free.fr, debyo.saptono@u-bourgogne.fr, charles.meunier@iut-dijon.u-bourgogne.fr, fanyang@u-bourgogne.fr

Résumé - Nous proposons dans cet article un outil de prototypage rapide des applications de traitement d'images sur des systèmes embarqués à base de FPGA. Le parallélisme intrinsèque au niveau des instructions a été automatiquement extrait à partir du code du source *C* (ou *C++*). Le modèle VHDL d'un processeur VLIW (Very Long Instruction Word) est généré et synthétisé pour le composant de type FPGA. L'objectif est de garder la souplesse de la programmation classique des processeurs et en même temps, de profiter des hautes performances du FPGA pour des applications en temps réel. En particulier, nous réalisons un modèle VHDL de processeur VLIW avec un jeu d'instructions variable et ciblé pour une application donnée. Ceci permet de réaliser le prototypage rapide sur le FPGA d'une manière optimale. L'approche a été testée et validée avec un ensemble d'algorithmes de traitement d'images basiques couramment utilisés, à l'aide d'un FPGA Virtex-6. Elle présente de multiples avantages : flexibilité, modularité, performance et réutilisation.

Abstract - This paper describes research result about enabling the VLIW processor model for real-time processing applications by exploiting FPGA technology. Our goals are to keep the flexibility of processors in order to shorten the development cycle, and to use the powerful FPGA resources in order to increase real-time performance. We present a modular VLIW VHDL processor model with a variable instruction set and a customizable architecture which allow exploiting intrinsic parallelism of a target application using advanced compiler technology and implementing it in an optimal manner on FPGA. Some common algorithms of image processing were tested and validated on an FPGA Virtex-6 based board using the proposed development cycle. Our approach applies some criteria for co-design tools: flexibility, modularity, performance, and reusability.

1 Introduction

Electronic embedded systems have an important role in real-time signal and image processing applications such as process control, telecommunication, satellites, and nowadays. Systems-on-Chip (SoCs) have become omnipresent because of the advances in design technology that make it possible to build complete systems containing different types of components on the same chip [1]. In this context, the FPGA, with its reconfigurability and easy integration capacity becomes a key solution for rapid prototyping of embedded systems, because using this electronic component, we are able to quickly create a rapid and fully functional prototype that can emulate and verify solutions, or even be embedded into the final system.

In order to design and synthesize FPGA based systems, a hardware description language such as VHDL or Verilog is necessary. Most hardware description languages are inherently concurrent and most high-level software languages are not considered trivial for non-hardware developers. One of the key factors that encourage the wide diffusion of electronic devices is the improvement of the man-machine interface, where the great challenge is to allow the use of complex electronic systems by software developers [2].

In this article, we propose an approach that targets rapid prototyping of signal and image processing applications on the FPGA. Firstly, algorithms are programmed in C as if they were to be executed on a classical processor. Then the advanced compiler OpenIMPACT [3] converts the original program into an independent language called *Lcode*. This intermediate representation provides Instruction Level Parallelism (ILP) which is analyzed and reorganized to Very Long Instruction Word (VLIW) instructions. Finally, a modular VLIW VHDL processor model with a variable instruction set is generated and implemented optimally with FPGA technology.

2 Application development cycle overview

Based on our previous works consisting of multi-DSP (Digital Signal Processor) implementation into FPGA devices [4], Figure 1 illustrates the recent approach composed of four stages. We use OpenIMPACT (Illinois Micro-architecture Project utilizing Advanced Compiler Technology) [3] to compile the original source code into an assembly intermediate representation "*Lcode*". This produced Lcode is optimized in ILP, for an Instruction Set Architecture (ISA). The primary/native ISA can be a parametric VLIW architecture. It admits processors of different composition and scale, especially with respect to the amount of parallelism offered. The customizable parameter space includes the number of clusters in a

multi-cluster processor, the make up of each cluster (types of functional units, the composition of the register files), and the instruction set including operation latencies and descriptors that specify when operands may be read and written, instruction format and resource usage behavior of each operation. The architecture instruction set is akin to the RISC load-store architecture, with standard arithmetic and memory operations.

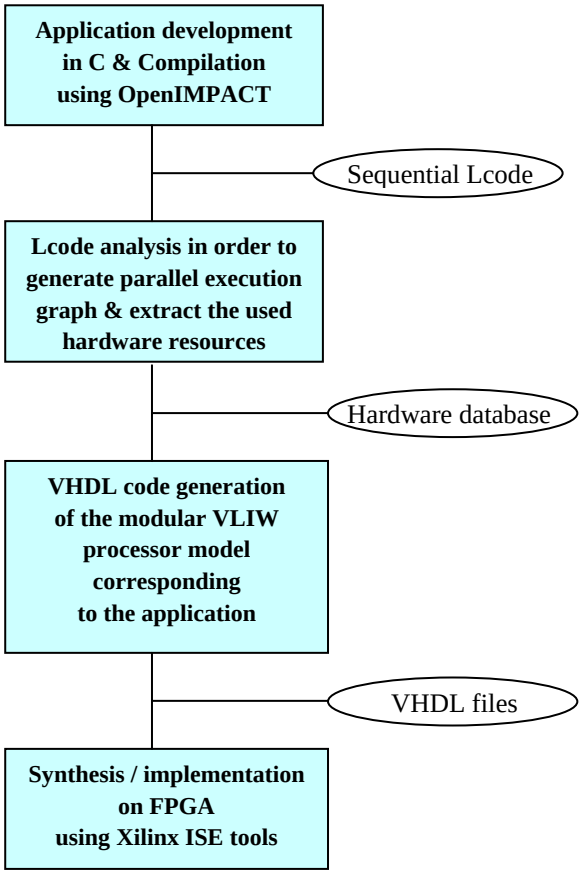

Figure 1: Proposed application development cycle.

We have developed a software generator which performs Lcode analysis in order to expose ILP and extract need used resources for a target application. This allows us to construct hardware resource database. The generator executes analysis in several phases. Firstly, it examines sequentially all instructions of Lcode and notes those can be executed in parallel with previous instructions (without data dependences). Then, it analyzes basic operation types and used registers of each parallel cycle and performs some optimizations (register number reduction, possible execution order change) respecting the consistency of the original Lcode. Finally, a parallel operation graph is generated in order to construct hardware resource database. Figure 2 displays the analysis and generator obtained results for the Sobel filtering processing. We can see that the main loop has be performed using 10 cycles after parallelization. Here, *Ri* (i=0-7) corresponds to a register and *ld* to a data loading from memory. The *lsl* represents data left shift. *C* indicates a constant. For example, at the step (cycle) 6, two operations have been simultaneously realized: **Add R0, R6 → R0,** and **Sub R2, R5 → R2.**

Using Lcode analysis and graph model generation results, a modular VLIW processor can be composed: for a target application, we construct its structure with know minimum need hardware resources and describe it in VHDL language (see section 3). It is noted that the VLIW processor is totally customizable and optimal for target application: number of functional units, operation types and used registers are just minimum necessary for give algorithm. Different pipelined levels of VLIW processors in order to control execution-time scheduling has been described in VHDL code and implemented onto an FPGA. All generated VHDL files for a target application have been synthesized using Xilinx ISE tools [5] to realize hardware implementation.

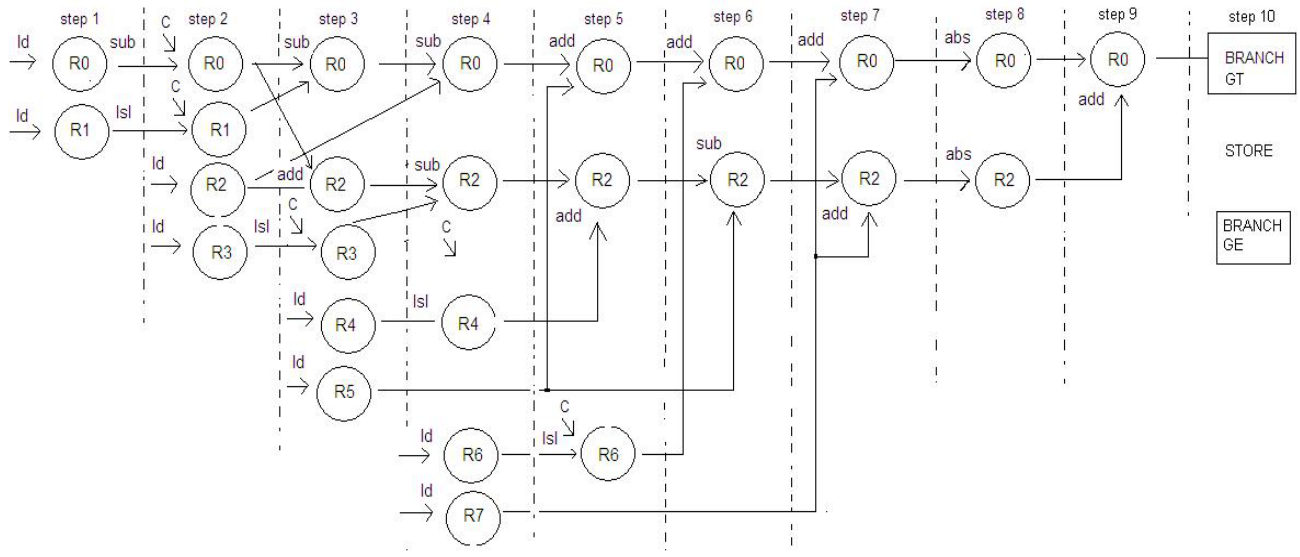

Figure 2: Operation graph generated by our software tool for the Sobel filter: each operation is realized using two operands from two registers and memorizes the result in a register.

3 Experiment results

In order to test and validate our development cycle of rapid prototyping, we have chosen three widely used low-level computer vision algorithms: filter Sobel for performing edge extraction, convolution with a mask of 3X3 and image erode. Our experiment results are obtained using an Intel Pentium 4 computer, with a clock speed of 2.8 GHz. The OpenIMPACT environment was pre-installed with Linux 2.6.22.5 kernel as underlying operating system and target hardware architecture is the Virtex 6-xc6vlx75T.

Figure 3 displays the modular VLIW processor architecture for the Sobel filtering application. In agreement with the Lcode analysis and graph model generation results; this modular VLIW processor is composed of 7 functional units (2 Add, 2 Sub, 2 Abs and 1 Shift left). For example, two operations have been simultaneously realized: **Add R0, R6 → R0,** and **Sub R2, R5 → R2** at the 6th cycle (see Figure 2). We have also generated a VHDL file to perform execution scheduling of the VLIW processor by multiplexer positioning (see **slct** signals in Figure 3).

2 arithmetic operators are triggered. The add(2) recuperates first operand (R2) from mux_add2_e0 and second (R6) from mux_add2_e1 and performs addition operation. The contain result is stored in the register R0. In the same manner, the sub(2) performs subtraction with two operands (R2 and R5) and transfers this subtraction result in the R2 register.

Based on the modular VLIW processor architecture, VHDL description is automatically generated. Synthesis and simulations of all VHDL code for the target algorithm are realized using the Xilink ISE tools. Table 1 shows Lcode analysis and parallelism extraction results for three basic image processing algorithms. Obtained experiments results concerning VHDL code synthesis and hardware simulation are respectively gives in Tables 2, 3 and 4. We can note that an average acceleration of 3**X** has been obtained for these three basic image processing using our graph model of parallel execution. In general, these algorithms use a small percentage of hardware resources available on FPGA: 0.16% of register Slices, 0.77% of LUT slices, and 73% of Block RAMs. These results have been obtained with an image size of 512x512 pixels and intern Block RAMs of FPGA are used in order to store original and resulting images.

We also made up hardware implementation performances comparisons between the DSP TMS C62 and the FPGA Xlink Virtex 6-xc6vlx75T using proposed application development cycle. These preliminary experiment results illustrate multiple advantages of this co-design tool: better processing speed performances, low level of consumption and more of flexibility using modular processor model.

Tab 1: Lcode analysis and parallelism extraction results.

Algorithm	Sobel	Conv.	Erode
Number of sequential cycle	27	14	64
Number of parallel cycle	10	9	14
Acceleration	2.7	1.6	4.6

Tab 2: Hardware implementation results for the Sobel filtering: Fr = 420 MHz.

Logic utilization	Used	Available	Ratio
Number of Register Slice	132	93120	0.14%
Number of LUT Slice	409	46560	0.88%
Number of Block RAMs	114	156	73%

Tab: 3 Hardware implementation results for the 3x3 convolution: Fr = 247 MHz.

Logic utilization	Used	Available	Ratio
Number of Register Slice	76	93120	0.08%
Number of LUT Slice	113	46560	0.24%
Number of Block RAMs	114	156	73%

Tab 4: Hardware implementation results for the image erode: Fr = 400 MHz.

Logic utilization	Used	Available	Ratio
Number of Register Slice	244	93120	0.26%
Number of LUT Slice	547	46560	1.2%
Number of Block RAMs	114	156	73%

4 Conclusions and perspectives

In this paper, we present a new co-design SW/HW approach for computer vision applications. The proposed development cycle allows non-electronic specialists to realize rapid prototyping of image processing. Since they can transform automatically their C codes in VHDL description for FPGA implementation in an optimal manner. Our method is based on advanced compiler technology and uses minimum necessary hardware resources for a target application thanks to modular VLIW processor model.

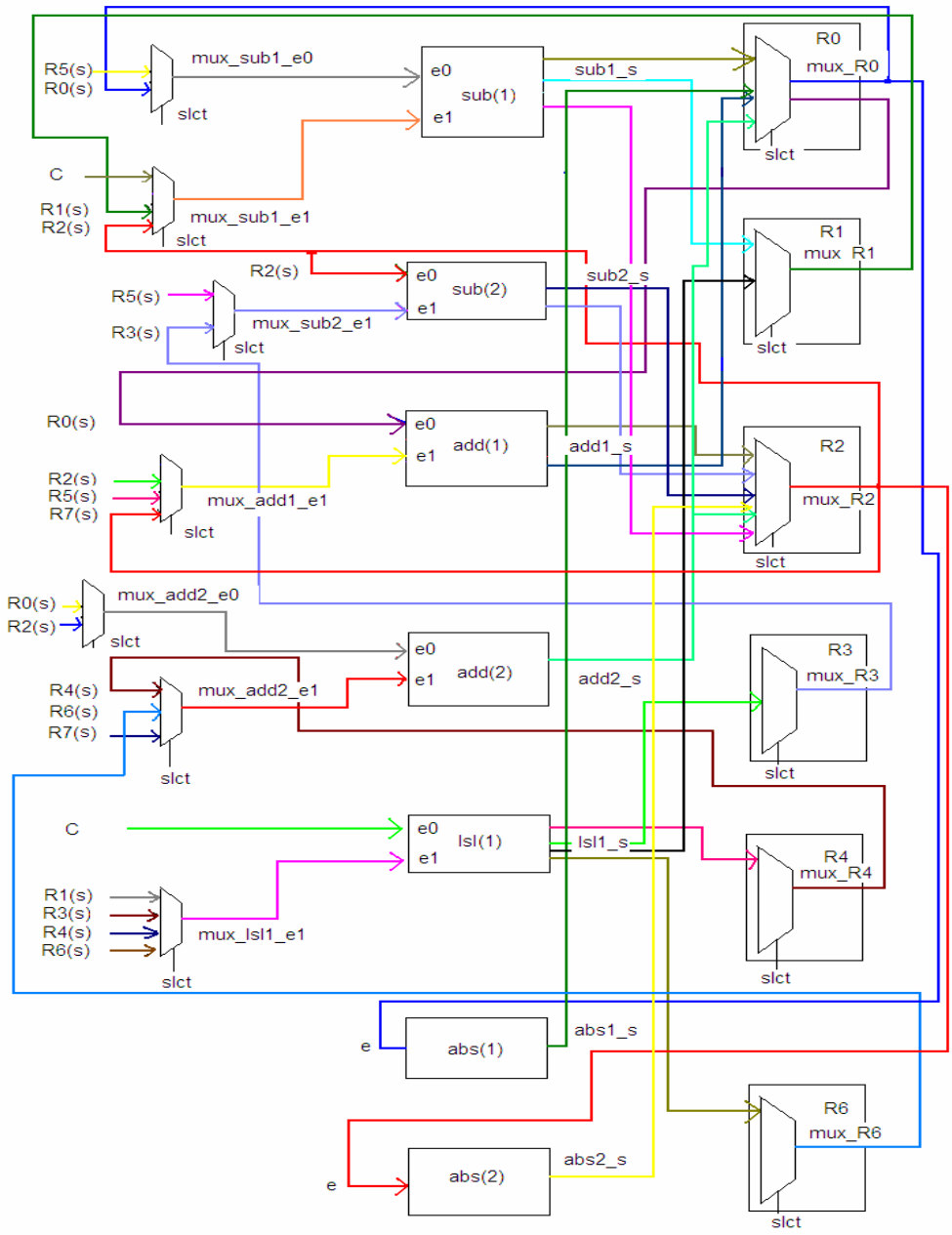

Figure 3: Modular VLIW processor structure for the Sobel filtering algorithm.

Our approach has been tested and validated using three common image processing algorithms: Sobel filter, Convolution 3x3 and image erode. In general, these algorithms use a small percentage of hardware resources available on FPGA and this allows considering others complexes post processing of image in the same FPGA device. Preliminary experiment results illustrate multiple advantages of this co-design tool: flexibility, modularity, performance, and reusability.

In perspectives, we want to implement multi-modular VLIW processors on FPGA in order to build SPMD (Single Program Multiple Data) machine. This allows greatly accelerating computation speed.

For application aspect, we want to test the proposed method for a complete chain of pattern recognition: palmprint processing for identity verification. This allows us to observe and evaluate behaviors of our tool for complexes and less regular applications.

References

[1] D. Gizopoulos, " Low-cost, on-line self-testing of processor core based on embedded software routines," *Microelectron. J.* 35, 443– 449, 2004.

[2] M. A. Aguirre, J. N. Tombs *et al.*, "Microprocessor and FPGA interfaces for in-system co-debugging in field programmable hybrid systems," *Microprocess. Microsyst.* 29 **(**2–3), 75–85, 2005.

[3] Gelato.org,"Impact Advanced Compiler Technology", http://gelato.uiuc.edu, 2009.

[4] V. Brost, F. Yang, M. Pandaivone, and N. Farrugia, "Multiple modular VLIW processors based on FPGA," *Journal of Electronic Imaging, SPIE,* 16(2):110, April -June 2007.

[5] ISE, 2009. ISE design suite 11.URL http://www.xilinx.com/tools/designtools.htm