SIMULINK AS A TOOL FOR PROTOTYPING RECONFIGURABLE IMAGE PROCESSING APPLICATIONS

B. Kovář, J. Schier

Ústav teorie informace a automatizace AV ČR, Praha

P. Zemčík, A. Herout, V. Beran

Ústav počítačové grafiky a multimédií
Fakulta informačních technologií
Vyskoké učení technické v Brně

Abstract

Image processing and computer vision algorithms become important in the industrial applications and in our daily life. The nature of many applications requires creation of standalone self-contained small systems capable of independent functionality. Our paper focuses on rapid prototyping and configuration tools for an embedded system. It will present novel concept of an image processing architecture based on interconnection of a programmable logical chip (FPGAs) with digital signal processor (DSP). In the paper, we shall outline a configuration tool tailored towards systems combining both DSP and FPGA, based on a configuration/programming scripting language and with a complex library of functions and modules for selected applications in signal and video processing. Also, design aspects of a Simulink based prototyping system with automatic conversion of block diagrams into the script language will be discussed.

1 Introduction

Image processing and computer vision methods become increasingly important not only in the industrial applications but also in our daily life. Image processing generally exploits tasks with very high computational demands. Such tasks can be handled by the "standard" processors and computers or by computers connected to the computational networks. However, such approach is not always suitable for various reasons (difficulties with programming, large dimension of the architecture, high consumption of energy, etc.). For these reasons, specialized hardware solutions based on digital signal processors (DSP) or a field programmable gate arrays (FPGA) are usually used in embedded systems.

The strong point of the DSPs is in their specialized architecture focused on multiply and accumulate operations (MACs) in which the current DSPs (e.g. Texas Instruments C64 architecture [1]) often outperform the "standard" processors. The DSPs at the same time have low energy consumption, easy to use architecture, and other nice hardware features; however, DSPs still suffer from the disadvantages of all the sequential processors, such as lack of massively parallel data processing, difficult bit manipulation, fixed data width, etc.

FPGAs, on the other hand, is design with fine-grain parallelism, witch makes it well suited for massively parallel algorithms. The weak points of FPGA are (with exceptions) in general relatively small on-chip memory capacity (which is important for image processing applications) and also from relatively narrow bandwidth memory interfaces, lack of wide-word processing units, and high cost of performing complex numerical operations, such as division, square root, logarithmic, exponential, and goniometrical functions. In smaller devices, these operations cannot be implement at all. Also, complex memory controllers and addressing units are difficult and expensive to implement.

The combinations of a DSPs and FPGAs are subject of research studies for several years already [2, 3]. Although the main features of the above mentioned architectures are suitable for

combination, development of applications for such combinations is generally difficult since it is needed to distribute the computational tasks between the processors and programmable logic; therefore, the application development support tools and methodology are probably as important as the potential of the architecture combining DSPs and FPGAs. This paper attempts to address both of the issues.

2 Architecture Overview

To achieve high performance in image processing applications, state-of-the-art components must be used. In the presented architecture components with best computational power/cost ratio are used – Texas Instruments C64xx series DSPs [1] together with the powerful Xilinx Virtes II FPGAs [4]. However, the architecture components and the development methods are generally applicable for the similar next generation of the DSPs and FPGAs. The proposed architecture

Figure 1: Photograph of the core computational module

consists of a a miniature "core computational module". The computational modules are placed on a PCI "motherboard" which provides their mutual interconnection. The design of the set of modules has been done jointly with Camea, spol. s r.o. (ltd.) who also manufactures all the modules. To exploit the features of the FPGA and DSP in the best feasible way, it was

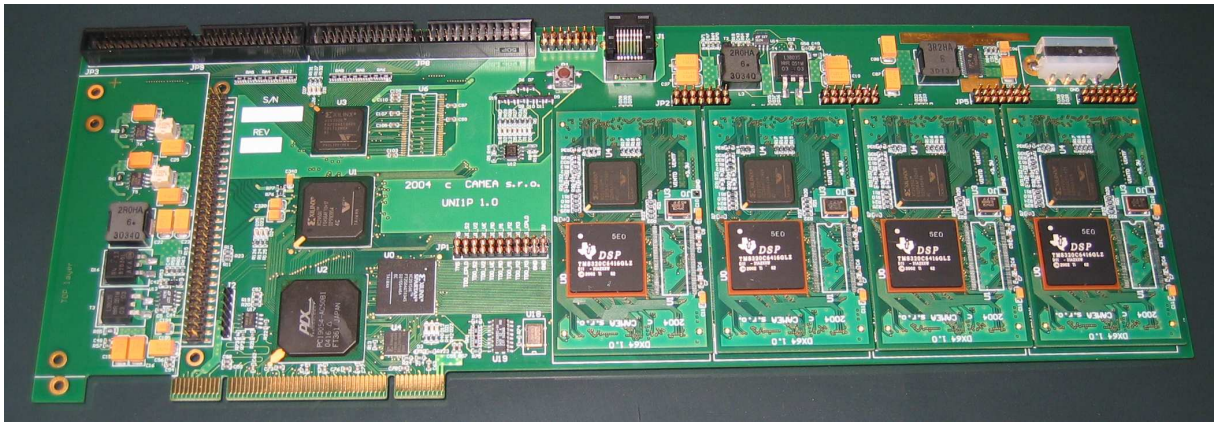

Figure 2: Photograph of the motherboard carrying four computational modules

decided that the FPGA will mostly rely on data transfers and data storage provided by the DSP. The FPGA is, therefore, connected to the peripheral bus interface of the DSP and is accessible as a set of registers in the DSPs memory space. Physically, the module is a small board with surface mounted electronic components. The module is thin to allow for itself and the motherboard to occupy only one PCI slot. The photograph of the system can be seen in Fig. 1. The motherboard to carry the four core computational modules contains merely a PCI interface, additional memory controller unit, and expansion and interface circuits. These circuits

are also built using FPGA chips so that interfacing of the modules to the motherboard is not too complex. The photograph of the actual motherboard can be seen in the Fig. 2. The physical layout of the board is a "full size PCI board" that occupies one PCI slot.

The basic setup of the motherboard and of the modules is done through a set of "C" functions that are available in UNIX (Linux) and Windows version. These functions provide means of PCI configuration, FPGA design upload, DSP software upload, etc. While these functions are specialized for the motherboard, the FPGA designs and DPS software they upload are generic and can be used in any configuration of the core computational modules.

3 Application Design

The standard approach to the application design would be to use the tools provided by the component manufacturers, such as Texas Instruments Code Composer Studio for the DSPs and Xilinx VHDL development tools for the FPGAs. Unfortunately, very little of the real-life image processing application designers are familiar with both of these tools. Additionally, the tools are rather expensive and difficult to use. Moreover, the application designers are not used to such tools at all. The idea of application development on the presented platform is

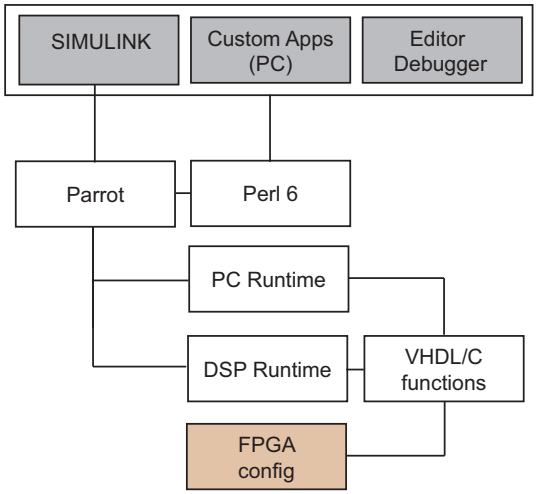

Figure 3: Software development modules

to allow the designers to keep the model of the application similar to what they are used to in standard computer programming. The algorithms are encoded as single or multi-threaded pieces of software using some kind of a procedural notation. The block diagram of the software development modules relationship is shown in Fig. 3.

The "standard" method of coding the application is in the Perl script language or alternatively in Parrot "assembly-language-like" intermediate code. Perl/Parrot code is compiled into a binary (byte-stream) form that is then executed in the DSP/FPGA system or in a PC. The binary code is accompanied with the function library which is implemented in C-language and some of its functions (the computationally demanding ones) in VHDL in order to be placed in the FPGA. Note, please, that while the application development is performed in Perl, the critical functions are written in "C" and optionally in VHDL.

The Perl was selected as the suitable choice for its open source nature, rather wide penetration, and also an efficient intermediate "assembly language like" code (Parrot, available in Perl version 6) whose runtime engine is efficient and also possible to modify in order to incorporate possible extensions of functionality in image processing, multi-thread application synchronization, etc.

4 Rapid Prototyping Tools

In some cases, the applications can be automatically generated. An example can be automatic block diagram transfer from Simulink into Perl using the Simulink TLC (Target Language Compiler tool) that can convert the block diagram to virtually any procedural language (Perl as well). While primarily targeted into the C-code, it can be configured to generate also other output. However, since the Simulink model is stored in a text file, virtually any general-purpose parser can be used. For some user-specific cases, specialized applications can be prepared for PC platform in order to allow the users to create applications e.g. through graphical interfaces. The applications designed in the Simulink are then automatically converted into Perl script (or directly into Parrot) and transferred into the target platform, potentially without the user knowing about the intermediate steps and about the script at all.

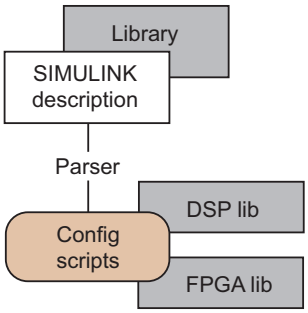

Figure 4: Parsing from Simulink to a configuration file

5 Conclusions

The presented system demonstrates the potential of the reconfigurable image processing architecture based on the combination of the DSPs and FPGAs for raster image processing. One of the major obstacles in usage of such systems complicated application development is addressed as well and the proposed solution is to use a scripting language for overall application description and "C" language and VHDL to code the library image processing functions. Although such approach is, of course, not ideal, it can be seen as an immediately useable approach that allows for application development at the current state of the art in computing.

Acknowledgments

This work has been supported by the Grant Agency of the Academy of Sciences of the Czech Republic under Project 1ET400750408.

References

[1] TMS3B0C6711, TMS320C6711B Floating point Digital Signal Processors, Texas Instruments, SPRS088B, USA, (available at http://www.ti.com)

[2] Zemčík P., Herout A., Crha L., Tupec P., Fučík O.: *Particle rendering pipeline in DSP and FPGA*, In: Proceedings of Engineering of Computer-Based Systems, Los Alamitos, US, IEEE CS, 2004, p. 361-368, ISBN 0-7695-2125-8

[3] Crha L., Fučík O., Zemčík P., Drábek V., Tupec P.: *Inter Chip Communicating System with Dynamically Reconfigurable Hadrware Support*, Poznaň, PL, 2003, p. 311-312, ISBN 83-7143-557-6

[4] Virtex-E 1.8 V Extended Memory Field Programmable Gate Arrays, Xilinx, DS025-2 (v2.2), USA, (available at http://www.xilinx.com)

Bohumil Kovář
Ústav teorie informace a automatizace AV ČR
Pod vodárenskou věží 4
182 08 Praha 8
Tel. +420-2 6605 2511
kovar@utia.cas.cz

Pavel Zemčík
Ústav počítačové grafiky a multimédií
Fakulta informačních technologií
Vysoké učení technické v Brně
Božetěchova 2
612 66 Brno
Tel. +420 5 4114 1217
zemcik@fit.vutbr.cz