RECONFIGURABLE IMAGE PROCESSING ARCHITECTURE WITH SIMULINK PROTOTYPING SUPPORT

J. Schier, B. Kovář

Ústav teorie informace a automatizace AV ČR, Praha

P.Zemčík, A. Herout, V. Beran

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

Abstract

A novel concept of an embedded image processing architecture is presented in the paper. This architecture is based on an interconnection of a programmable logical chip (FPGA) with a digital signal processor (DSP). Both these devices have characterics that complement well each other for broad-class of data-intensive image- and video-processing tasks.

For efficient utilization of such device, a multi-level configuration system is needed. At the low level, it has to provide means for hardware setup and run-time re/configuration of the device. At high level, tools for rapid application development are required.

The paper describes both the hardware and software architecture of the system and the configuration methods utilized for application development for the proposed architecture.

1 Introduction

Image processing is, in general, characterized by very high computational demands. Although it can be handled by "standard" computers, such solution is not viable for an embedded system, where dimensions of the computer system, power consumption or data throughput are of concern. For these reasons, specialized hardware solutions based on a digital signal processor (DSP) or a field programmable gate array (FPGA) are usually used in embedded systems.

The architecture of a DSP processor is tuned for fast multiply and accumulate operations (MACs). In the same time, the DSP is characterized by low energy consumption, high throughput I/O subsystem and some degree of parallel processing.

The architecture of FPGA, on the other hand, is designed with fine-grain parallelism, which makes it well suited for masivelly parallel algorithms. The weak points of FPGA are (with exceptions) relatively small capacity of the on-chip memory and relatively narrow troughput of 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 implemented at all). Also, complex memory controllers and addressing units are difficult and expensive to implement.

In the recent years, a combination of a DSP with FPGA for advanced image rendering tasks has been studied extensively [1, 2]. However, development of applications for such combination is rather 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 the processors and programmable logic. This paper attempts to address both of the issues.

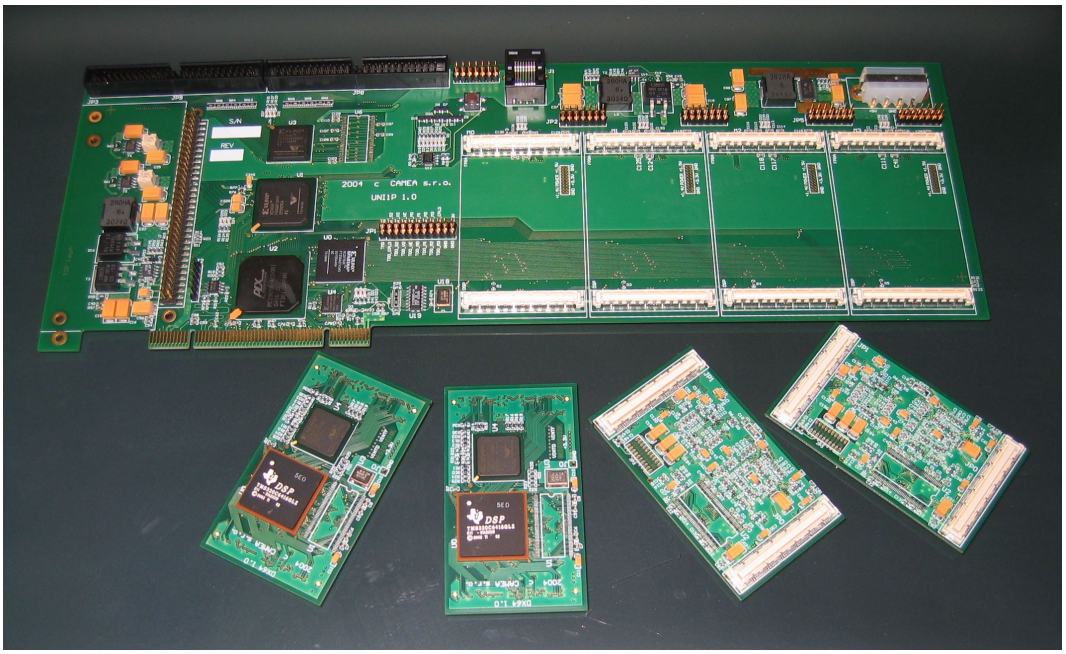

Figure 1: PCI carrier with computational modules

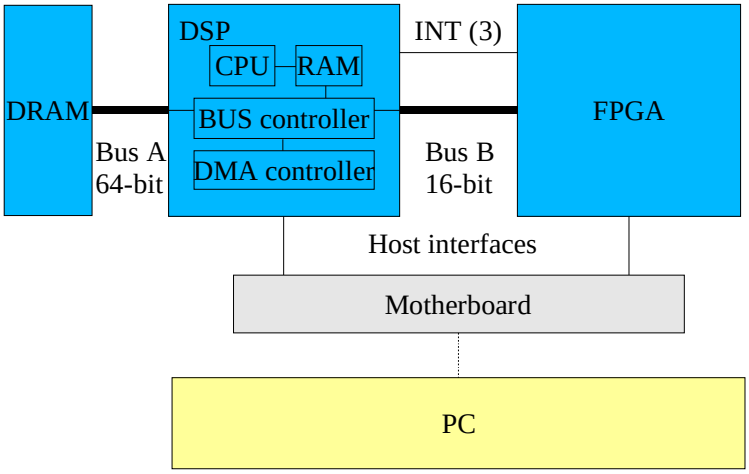

Figure 2: Block scheme of the system

2 Example of an image processing architecture

In this section, an example of an image processing architecture, produced by Camea s.r.o., which is designed specifically for embedded high performance computations, will be presented. It consists of computational modules, carried by a PCI carrier-board that provides also the necessary interconnections, connection to the host computer and interfaces for optional application-specific modules (see Figure 1). One such PCI board can hold up to four modules. The modules contain high performance Texas Instruments C64xx series DSP and Xilinx Virtex II FPGAs.

The FPGA device relies on the data transfers and data storage provided by the DSP. It is connected to the peripheral bus interface of the DSP and is accessible as a "set of registers" in the memory space of DSP: the C64xx series of DSPs have two external buses dedicated for memory and peripheral interface. The FPGA is connected to the peripheral bus so that it does not affect the memory bandwidth of the system and the DSP's raw computational speed. The data delivery to and from the FPGA is accomplished through the DMA controllers built in the DSP (see Figure 2).

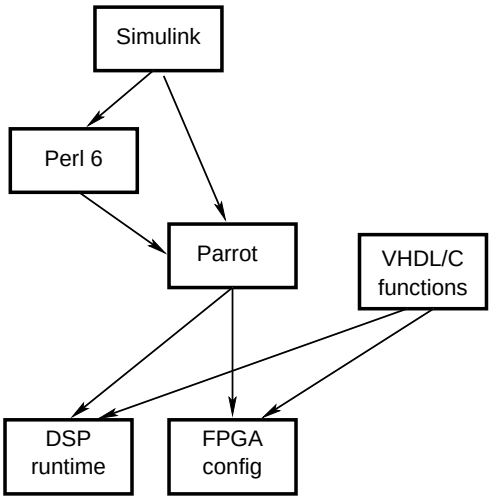

Figure 3: Design flow of the system

The basic setup of the motherboard and of the modules is done through a set of 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 idea of application development in our project is to shield the designer from the complexity of specialized hardware development tools and to provide him with an environment similar to what he is used to in standard 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 Figure 3.

An application is set-up in the form of a Perl configuration script or, alternatively, in Parrot "assembly language-like" intermediate code which calls/loads library functions and modules. The Perl/Parrot code is compiled into a binary (byte-stream) form that is then executed in the DSP/FPGA system. Hence, while the application development is performed in Perl, the critical functions are written in C and/or in VHDL.

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

4 Rapid prototyping tools

For application prototyping, a blockset is being developed for use with Simulink. The blocks that it will contain will correspond with the counterparts in the C/VHDL library. An appropriate tool for automatic parsing of the Simulink design into a configuration file is being investigated (see Figure 4). The standard option for Simulink the Target Language Compiler. While primarily targetted 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, such as lex/yacc, can be used. The advantage of the later solution may be that lower cost (no additional license

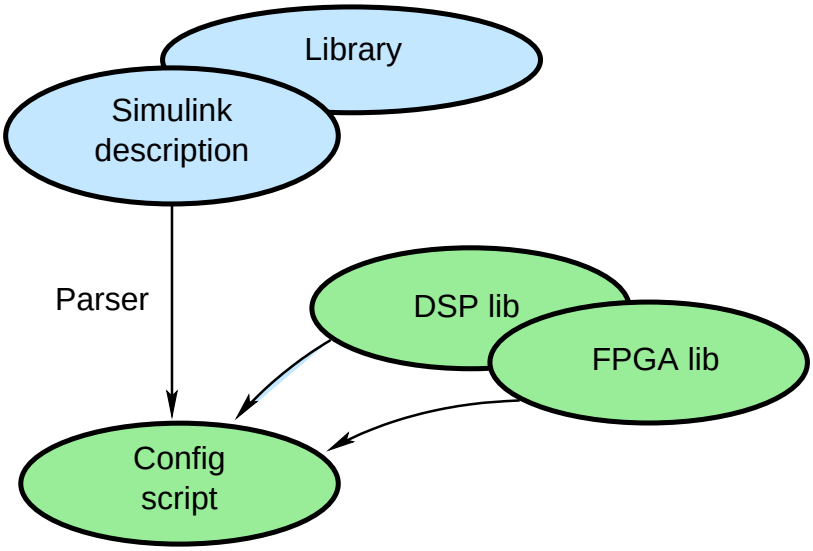

Figure 4: Parsing from Simulink to a configuration file

is required) while it may be sufficient for the purpose of preparing the configuration file.

5 Ackowledgement

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

References

[1] A. Herout, P. Zemčík, V. Beran, and J. Kadlec. Image and video processing software framework for fast application development. In *AMI/PASCAL/IM2/M4 workshop*, Martigny, 2004.

[2] P. Tišnovský, A. Herout, and P. Zemčík. Cache-based parallel particle rendering engine. *ElectronicsLetters.com*, 2003(1), 2003. ISSN 1213-161X.

Jan Schier
Ústav teorie informace a automatizace AV ČR
Pod vodárenskou věží 4
182 08 Praha 8
Tel. +420-2 6605 2470
schier@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