SIMULINK MODEL CONVERTER FOR EMBEDDED VIDEO ACCELERATOR

B. Kovář, J. Schier

Ústav teorie informace a automatizace AV ČR, v.v.i., Praha

P. Zemčík, A. Herout, J. Zuzaňák

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

Abstract

Image processing and computer vision algorithms have become important both in the industrial applications and in the consumer applications of a our daily life. Because of required data throughput and processing flexibility, an architecture based on processors (DSP, processor core, etc.), connected with an accelerator (programmable logical chip - FPGA) is sometimes advantageous.

A scripting language-based configuration engine for such an embedded system is described and the design aspects of converting Simulink block diagrams into this scripting language are discussed in the paper.

We outline a configuration tool tailored towards system combining both DSP and FPGA, based on configuration/programming scripting language and with a library of functions and modules for selected applications in signal and video processing. Regarding to limitations of the RIPAC[1] target platform, the data representations and possible applications, we decided to prepare also our own video and image I/O blockset.

1 Introduction

Image processing applications and computer vision methods become increasingly important both in industrial applications and but in the consumer appliances of our daily life. Machine vision is typically characterized by very high computational demands. While such demands can be handled by standard computer or by a set of networked computers, such a approach is not always suitable for various reasons (difficulties with programming, large dimension of the architecture, high level of energy consumption, etc.). For these reasons, specialized hardware architectures based on programmable logic (FPGA - Field Programmable Gate Array) or on an architecture combining a processor (DSP, processor core, . . .) connected with an accelerator (FPGA) is sometimes advantageous for embedded systems. Let us note, though, that with the recent advances in multicore low-power general-purpose processors and small-factor PCs, the division becomes more hazy.

The DSPs have a nice hardware features – very fast multiple and accumulate instructions, low energy consumption and easy to use architecture. On the other hand, they 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. On the contrary, FPGAs are designed with fine-grain parallelism features, witch makes them 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.

[1]Rapid prototyping tools for development of HW-accelerated embedded image- and video-processing applications – GA AV grant agency project 1ET400750408

The combinations of a DSPs and FPGAs are subject of research studies for several years already [1, 2, 3, 4]. 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. A scripting language-based configuration engine for such an embedded system is described and the design aspects of converting Simulink block diagrams into this scripting language are discussed in the paper. We outline a configuration tool tailored towards system combining both DSP and FPGA, based on configuration/programming scripting language and with a library of functions and modules for selected applications in signal and video processing.

Regarding to limitations of our target platform, the data representations and possible applications, we decided to prepare also our own video and image I/O blockset.

2 Architecture Overview

The system used in our project is based on the Texas Instruments C64 series DSPs [5], linked with the Xilinx Virtex II FPGAs [6]. At the time of the project beginning, these components were used for favourable computational power/cost ratio. However, the architecture components and the development methods are generally applicable for the similar next generation of the DSPs and FPGAs. The proposed architecture consists of a a miniature "core computational module".

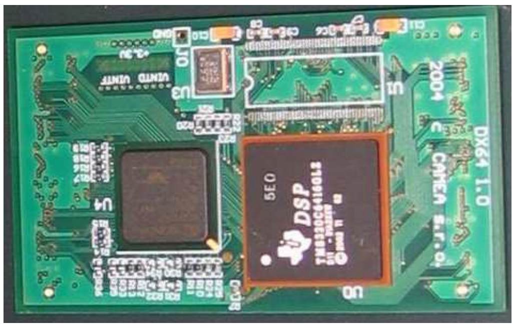

Figure 1: Photograph of the core computational module

The computational modules are placed on a PCI "carrierboard" which provides their mutual interconnection. Both the carrier and the modules were developed and manufactured by Camea, s.r.o. (Ltd.).

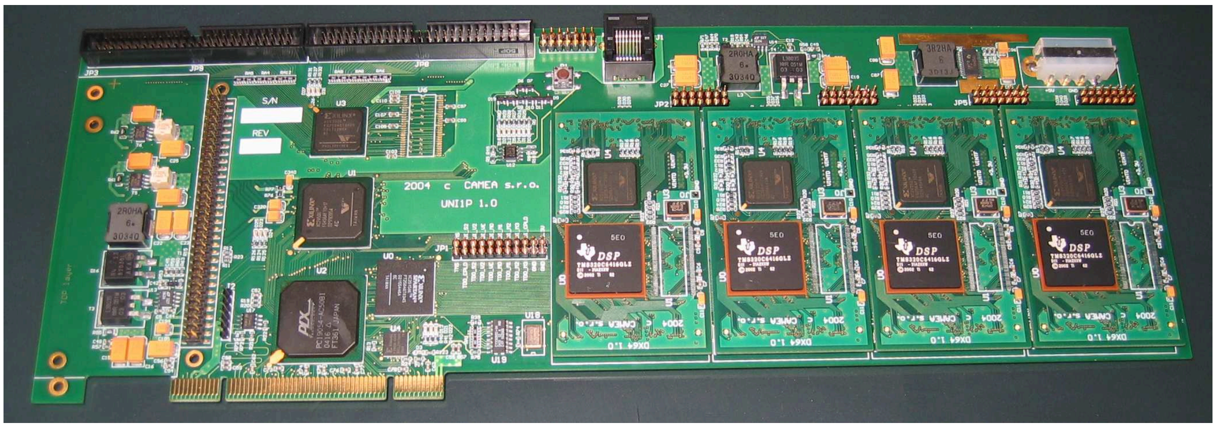

Figure 2: Photograph of the motherboard carrying four computational modules

To exploit the features of the FPGA and DSP in the best possible way, it was 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 memory space of the DSP. Physically, the module is a small board with surface mounted components (SMC-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 carrierboard, which can carry up to four modules, contains the PCI interface circuitry, additional memory controller unit, and the 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 motherboard is presented in 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

Big problem that the designer faces very often is how to pass from the algorithmic design to its physical implementation. 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, only few application designers developing real-life image processing applications are familiar with these tools. Moreover, the tools are rather expensive and their efficient use is not trivial. Also, the application designers are rather used to use either high-level design and simulation tools or standard C programming tools (GNU-C, Visual-C), combined with C-based image processing libraries when developing new product.

3.1 Design with Simulink

One of the most commonly used high-level tools is Matlab and Simulink. It allows the designer to put together a structural simulation very easily and quickly checking the algorithm or making the necessary modifications to it. Working directly with any low-level implementation tool from the start is simply not practical. Every small change in the algorithm may sometimes require partial redesign of the whole implementation. Therefore an automatic link between the high-level algorithmic design, like Simulink model, to some low-level implementation description, like a VHDL or pre-compiled modules, would lead to great effort and time savings in the design cycle.

The Simulink was chosen as the high-level design tool for two reasons. In the first instance, the description of a common Simulink block is a quite similar to netlist of the physical implementation. Secondly, the fact that Simulink makes it possible to design both behavioral and structural designs just confirms our choice.

Unfortunately, the support for image processing and computer vision in the Simulink without extensions (e.g. Video and Image Processing Blockset) is very limited. Regarding to limitations of our target platform, the data representations and possible applications, we decided to prepare our own video and image I/O blockset. Current version is capable of reading, writing and displaying of image sequences stored on a local drive. Video files are directly supported by Matlab. Each I/O block contains data conversion into appropriate format which is compatible with hardware implementation. The blocks that perform image processing and computer vision operations will be later included into the library of standard Simulink entities.

3.2 Simulink converter

The task of the converter utility is to transform the design the high-level Simulink description to the configuration language for low-level implementation, to be interpreted by an embedded interpreter.

The tasks of the converter can be defined as follows:

1. Analyze Simulink model and identify:
   - common and user defined blocks and sub-systems,
   - connections between blocks (and ports for multilevel models),
   - blocks functions and parameters.
2. Based on collected information generate configuration language.

The configuration language has similar syntax rules as ANSI C/C++. Regarding to hardware implementation, system model is analysed and the interconnected block sets are divided into two groups: serial blocks and paralel blocks. These new structural properties are implemented using following instructions:

```
b0 = serial_block {                b1 = paralel_block {
  function_1();                      function_1();
  function_2();                      function_2();                      ]
  ..                                 ..
  function_n();                      function_n();
};                                 };
```

Each block is identified by its own index, e.g. `b0`. Block is then executed using command `execute_block(b0)`. It is possible to use paralel blocks inside serial block and vice versa. The syntax of functions inside each block is similar to C/C++ language.

```
f_output = f_input.function(parameter_1, parametr_2, ., parametr_n));
```

A simple Simulink scheme with the corresponding description in the configuration language is described in Fig. 3.

```
b0 = serial_block {
         c0 = pipe_from_file("input");

         paralel_block {
                c3 = c0.op_max();
                c5 = c0.op_sin();
                c2 = c0.op_min().op_gain(5);
         };

         paralel_block {
                c6 = c2.op_sub(c3).op_add(c5);
         };

         c6.to_file("output");
};

execute_block(b0);
```

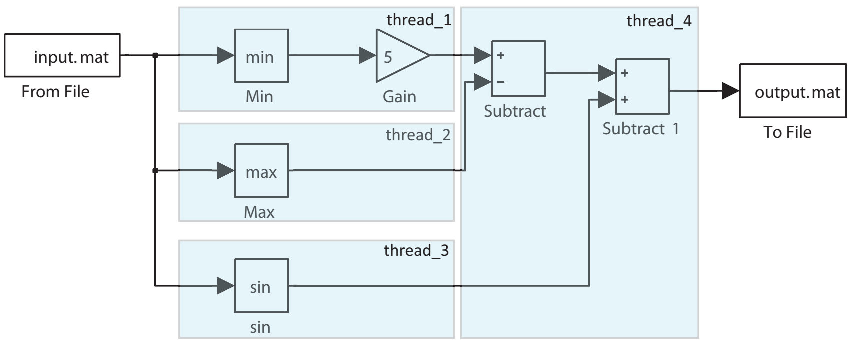

Figure 3: Simple Simulink model used for demonstraion

In order to convert the Simulink scheme into the scripting language, the following steps have to be taken:

- Determine blocks used in the model:
  - Assign an index number to each block name
- Determine links between blocks
  - Assign an index number to each link
  - Determine the link names, if used
- Determine, which blocks are the source blocks, which are the sink blocks, and which blocks are the intermediate blocks.
- Determine the data dependences and possible parallelism among the (groups of) blocks
- Using the information on the syntax of the scripting language, write out the script for the system configuration.

4 Conclusions

An image processing architecture for raster image processing, based on a combination of the DSP and FPGA, was outlined in the paper. One of the major obstacles in usage of such systems — complicated application development — is addressed and the proposed solution is to use a configuration language for overall application description and the C- and VHDL languages to code the library of image processing functions. The conversion of the Simulink schemes to the scripting language has been discussed briefly.

It should be noted that the proposed Simulink converter is not intended to replace sophisticated tools such as VHDL coder or System Generator for DSP, rather to serve as a simple-to-use tool in one well-defined, specific application field.

Acknowledgments

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

References

[1] E. A. Hakkennes and S. Vassiliadis: *Multimedia Execution Hardware Accelerator*, In: Journal of VLSI Signal Processing Systems, 28, p. 221-234, July 2001

[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] E. Moscu Panainte, K.L.M. Bertels, S. Vassiliadis: *The Molen Compiler for Reconfigurable Processors*, In: ACM Transactions in Embedded Computing Systems (TECS), February 2007, Volume 6 , Issue 1

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

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

Jan Schier, Bohumil Kovář
Ústav teorie informace a automatizace AV ČR
Pod vodárenskou věží 4
182 08 Praha 8
Tel. +420-2 6605 2511
`schier,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`