14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

THE RAPID PROTOTYPING EXPERIENCES OF IMAGE PROCESSING ALGORITHMS ON FPGA

V. Brost, F. Yang, M. Paindavoine

Laboratoire Le2i, Aile de l'Ingnieur - Mirande
Universit de Bourgogne, BP 47870 - 21078 DIJON cedex, France
Tel : +33 3 80 39 36 08 Fax : +33 3 80 39 59 10
vincent.brost@free.fr, fanyang@u-bourgogne.fr, paindav@u-bourgogne.fr

ABSTRACT

Recent FPGA chips, with their large capacity memory and reconfigurability potential, have opened new frontiers for rapid prototyping of embedded systems. With the advent of high density FPGAs it is now feasible to implement a high-performance VLIW processor core in an FPGA. We describe research results of enabling the DSP TMS320 C6201 model for real-time image processing applications, by exploiting FPGA technology. The goals are, firstly, to keep the flexibility of DSP in order to shorten the development cycle, and secondly, to use powerful available resources on FPGA to a maximum in order to increase real-time performance. We present a modular DSP C6201 VHDL model which contains only the bare minimum number of instruction sets, or modules, necessary for each target application. This allows an optimal implementation on the FPGA. Some common algorithms of image processing were created and validated on an FPGA VirtexII-2000 multimedia board using the proposed application development cycle. Our results demonstrate that an algorithm can easily be, in an optimal manner, specified and then automatically converted to VHDL language and implemented on an FPGA device with system level software.

1. INTRODUCTION

Electronic embedded systems play an important role in diverse real-time signal and image processing applications such as process control, telecommunication, satellites, and the medical field. In the last 10 years, increasing technological capacity has led to the emergence of the System-on-Chip (SoC). Nowadays, SoCs have become ubiquitous 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][2]. In this context, the FPGA (Field Programmable Gate Array), with its reconfigurability and high integration capacity becomes a key solution for rapid prototyping of embedded systems[3]. These user programmable solutions are capable of performing the hardware part of a design for a significantly lower price and maintain many of advantages of the ASIC (Application Specific Integrated Circuit) solutions. The most interesting characteristic of FPGA, with its reconfigurable nature, is probably the ability to quickly create a rapid and fully functional prototype that can emulate and verify solutions, or even be embedded into the final system.

In the standard FPGA based prototyping methodology, algorithms are first developed in standard software programming languages such as C or Matlab on a personal computer or workstation. When the algorithm is later implemented in hardware, the C (or Matlab) code is translated into a hardware description language such as VHDL or Verilog. Finally, the design is synthesized for an FPGA-based environment where it can be tested[4]. Most hardware description languages are inherently concurrent in nature and most high-level software languages are not trivial for non-hardware developers. One of the key factors that encourages 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 system by software developers. Many research laboratories and industrial manufacturers focus their effort in this way.

In this article, we propose a method in order to implement Digital Signal Processors (DSP) based on the FPGA for real-time signal and image processing. The DSP TMS320 C6201 is a Very Long Instruction Word (VLIW) processor[5] capable of issuing and executing multiple operations per cycle, bundled in a "MultiOp" long word instruction. The processor is equipped with multiple functional units so as to exploit the parallelism that has been exposed by the compiler. With our approach, algorithms are programmed in C as if they were to be executed on a DSP. The code is analyzed automatically, and a DSP VHDL model with a variable instruction set is generated and implemented with FPGA technology. We easily exploit the parallelism of target applications using the VLIW processor compiler, and quickly implement corresponding functional units on the FPGA using a DSP C6201 VHDL model generator. The most innovative aspect of our work resides in the concept of modular VLIW processor model.

Experimental results allow us to observe a phenomenon of under-utilization of DSP. For example, all hardware resources are not inevitably necessary for a target application, or they are not all used at

the same time. For this reason, we show that it is possible, by analyzing the instruction code, to realize modular DSP C6201 VLIW processors using available hardware resources in an optimal manner. We enable a CPU model of the VLIW DSP on the FPGA which contains only the bare minimum number of instruction sets, or modules, necessary for each target application. Thus, several modular VLIW processors can easily be made with a single FPGA in order to build a parallel machine.

In following sections, we introduce firstly a short description of the TMS320 C6X DSP CPU architecture. Then principles and concepts of our approach are illustrated by an edge extraction example using the Sobel filter. We describe also experimental results obtained for some common algorithms of image processing realized on a FPGA Virtex II-2000 multimedia board using the proposed development cycle.

2. TMS320 C6X DSP CPU DESCRIPTION

In recent years, diverse DSP architectures have been developed in order to boost processor performance. These processors achieve higher performance mainly by increasing their parallelism level and clock rate. The TMS320 C6X DSP of Texas Instruments (TI) employs an advanced VLIW processor architecture. TMS320 C6x DSP CPU frequencies range from 150 MHz to 1 GHz. The DSP C6x possesses 8 pipelined levels of fetch packets and can do eight operations per clock-cycle[6]. Their instruction length is 32 bytes. Figure 1 shows the CPU architecture of the TMS320 C6201.

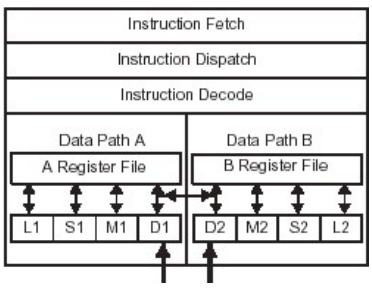

Figure 1: The TMS320 C6201 CPU includes a program fetch unit, instruction dispatch and decode units, two data paths supported by eight functional units, and 32 general purpose registers.

The instruction processing occurs in each of the two data-paths, each of which contains four functional units (L1, S1, D1, and M1 for data-path A and, L2, S2, D2, and M2 for data-path B) and 16 32-bit register files (RA0-15 for data-path A and RB0-15 for data-path B). Each functional unit has access to all registers of its data-path and registers of the other data-path via two cross-paths. It is possible to exchange data between memory and registers. Each data-path possesses 93 operation types : 23 for unit L, 28 for unit S, 20 for unit M, and 22 for unit D.

3. PROPOSED APPLICATION DEVELOPMENT CYCLE PRESENTATION

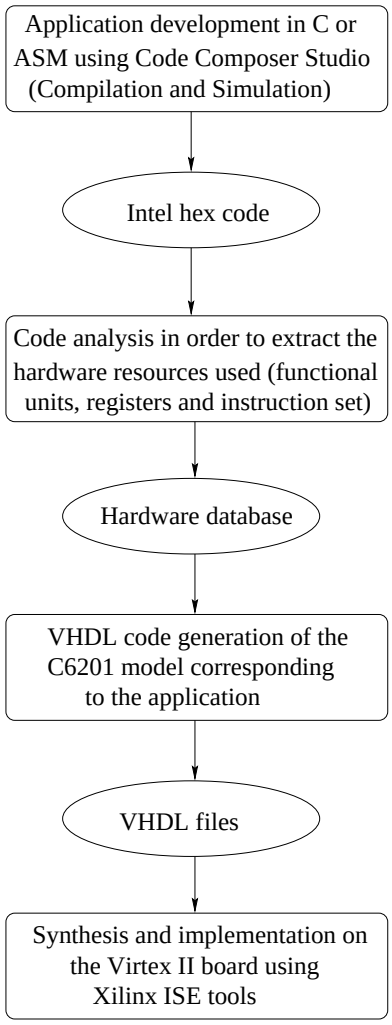

Figure 2: Application development cycle: the VHDL generator automatically produces only description code corresponding to hardware resources used for a target application.

The Figure 2 presents our application development cycle. We describe its principles and concepts from algorithm programming to implementation on an FPGA, by an edge extraction example using the Sobel filter which is usually implemented with a convolution using two 3×3 masks.

3.1 Application code analysis

For a target application, we can develop applications in C or Assembler as if they will be executed on a DSP. Then, the corresponding C6201 machine code is generated using Code Composer Studio compiler of TI. For DSP C6201, VLIW instruction length is 32 bytes whose 8 segments can be destined to a single functional unit (sequential) or up to eight functional units (parallel). Each segment of the C6201 code contains information as a).

Operation type, b). Functional unit where the operation will be executed, c). Used registers for operation sources and operation destination, and d). Operation will be or will not be executed in parallel with others.

By analyzing the complete C6201 code for an algorithm, we can know the number of functional units used and their instruction set, as well as necessary registers. This stage of hardware resource extraction is automatically done by our generator (written in Visual C++). The hardware resource database is composed of two parts: a database for each functional unit of C6201 CPU, and a common database for registers of C6201 CPU.

The behavior of each C6201 functional unit is organized into operation groups, with each group containing a set of operations having some common characteristics. Each operation is described in terms of its opcode and operands. Each operand is classified either as source or as destination. In the first part of the database, we extract source operand datapaths and corresponding opcode for all operations destined to each functional unit. In the second part of the database, we arrange registers used for the destination operands of each functional unit.

From the hardware resource database, we can regroup all analysis results (see tables 1, 2, and 3) in order to obtain necessary information for the modular C6201 CPU. For this Sobel filter application, we use only 6 functional units (8 available in C6201 CPU), 23 registers (32 available in C6201 CPU), and 16 operation types (186 available in C6201 CPU). We also economize 81.8% (70/384 used) of connections for source operand datapaths and 75% (40/160 used) of connections for destination operand datapaths.

Table 1: *Summary analysis results of operation type for functional units used. The ratio represents the number of necessary operation types in comparison with the number of available operation types in the C6201 CPU.*

Unit	Opcode	Ratio
L1	SUB, ABS, ADD	3/23
L2	AND, ADD, SUB, ABS	4/23
S1	ADD, Shift, AND	3/28
S2	ADD, Shift	2/28
D1	ADD, SUB	2/22
D2	ADD, SUB	2/22
M1		0/20
M2		0/20

3.2 VHDL generation of the CPU model

From the hardware resource used database established in the code analysis stage, our generator makes a minimum C6201 CPU model for a target application. Figure 3 shows the general structure of generated C6201 CPU VHDL model corresponding to a complete application. Each functional unit retrieves two source operands src1 and src2 and undertakes an operation. Then the destination operand is transmitted to a register for memory storage or in order to prepare for another operation.

Table 2: *Summary analysis results of source operand datapaths for functional units used. The connection ratio represents the number of necessary connections between registers and functional units for sources operand datapaths in comparison with the number of available connections in the C6201 CPU.*

Functional unit	Connection Src1	Ratio Src2
L1	3/32	7/32
L2	3/32	8/32
S1	6/16	10/32
S2	2/16	11/32
D1	4/16	5/16
D2	5/16	6/16
M1	0/16	0/32
M2	0/16	0/32

Table 3: *Summary analysis results of destination operand datapaths for registers used. The connection ratio represents the number of necessary connections between registers and functional units for destination operand datapaths in comparison with the number of available connections in the CPU C6201.*

Register A	Connection ratio	Register B	Connection ratio
RA0	1/5	RB0	2/5
RA1	3/5	RB1	1/5
RA2	3/5	RB2	2/5
RA3	0/5	RB3	2/5
RA4	2/5	RB4	0/5
RA5	0/5	RB5	1/5
RA6	1/5	RB6	3/5
RA7	0/5	RB7	0/5
RA8	1/5	RB8	1/5
RA9	3/5	RB9	2/5
RA10	2/5	RB10	0/5
RA11	0/5	RB11	1/5
RA12	2/5	RB12	0/5
RA13	0/5	RB13	1/5
RA14	2/5	RB14	1/5
RA15	2/5	RB15	1/5

We respect eight pipelined levels of the DSP C6201 in order to control execution time scheduling: program fetch, instruction dispatch, and instruction decode units deliver up to eight 32-bit instruction to functional units every CPU clock cycle. This mechanism has been described in VHDL code and implemented on a FPGA.

Note that the CPU VHDL model is modular. VHDL codes are generated automatically using VLIW code analysis results specific to a target application. Therefore, we can economize FPGA hardware resources so as to implement several modular C6201 CPU models.

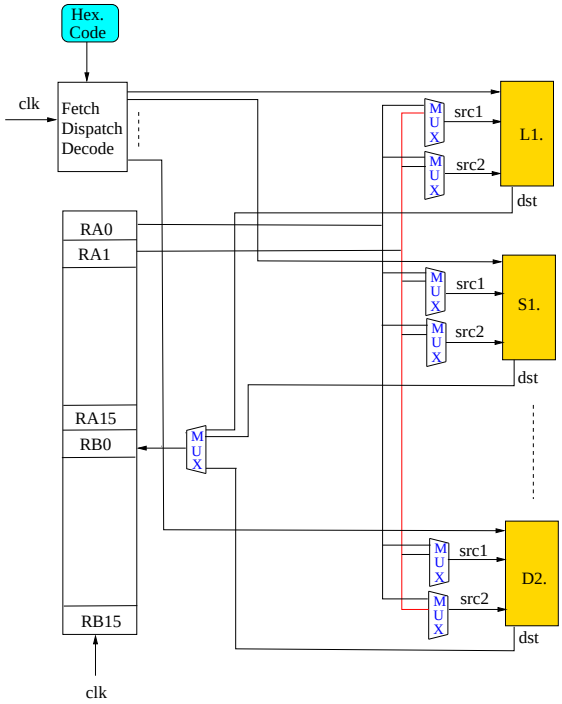

Figure 3: Simplified diagram of C6201 CPU VHDL model: we draw only the source operand datapaths from registers RA0 and RA1, and only the destination operand datapaths are represented for register RB0.

3.3 Multi-DSPs implementation on a FPGA

Complete application design cycle has been realized and validated on an embedded system containing an FPGA VirtexII-2000. First, we implemented the Sobel filter with a single model of DSP TMS320 C6201. This first hardware implementation used 18% of the available slices (slice = basic cell logic of FPGA) which is quite low. Then, we embedded four models of the DSP TMS320 C6201 in order to improve hardware implementation performance. Each 512×512 pixel image is divided into 4 horizontal bands, each of which is processed by a DSP C6201. Thusly, we built a SPMD (Single Program Multiple Data) parallel machine. Table 4 gives hardware performances. Only 4 ms are necessary for edge extraction of an image using Sobel filter (Oscillator frequency = 100 MHz). This leaves time to process the other, more complex, stages.

Table 4: *Hardware implementations performances of the Sobel filter.*

	1 DSP	4 DSPs
Number of Slices used	1965	6544
Slices used ratio	18%	60%
Processing speed	15.7 ms	4 ms

4. EXPERIMENT RESULTS

In this section, we present obtained experiment results for rapid prototyping of image processing applications on the FPGA VirtexII using the proposed development cycle.

The architecture used for this research project is the multimedia board from Xilinx (see Figure 4). This contains a VirtexII XC2V2000 as the user application FPGA where we implement modular C6201 CPU VHDL models and one PICOBLAZE VHDL model[7] for control management. This board also supports five independent banks of $512K \times 36$ bits 130 MHZ ZBT RAM with byte write capability. This memory may be used as microprocessor code/data store, or as video frame buffers. Real time video is supported with a PAL-NTSC video decoder/encoder pair. The VirtexII also generates the digital video stream for the video encoder. Video output is provided in S-video and composite video as well as RGB formats.

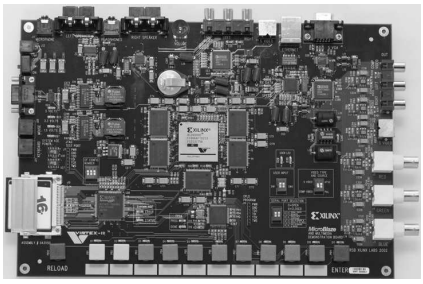

Figure 4: The VirtexII multimedia board.

We have created and tested some common image processing algorithms using this multimedia board. Obtained experiment results are presented in tables 5, 6 and 7. Table 5 gives results of code analysis and hardware resource extraction for each algorithm : operation type used ratio, source (*src*) connection ratio and destination (*dst*) connection ratio between functional units and registers of two datapaths. Note that Convolution, Median and Erosion operations use masks of 3×3, and IIR filter operation has been realized with 9 coefficients.

Table 5: *Results of code analysis and hardware resource extraction for each modular VHDL DSP. The DSP TMS320 C6201 CPU possess 186 operation types, 384 connections for source operands and 160 connections for destination operands.*

Algorithm	Operation type used ratio	Src connec. ratio	Dst connec. ratio
Conv.	32/186	96/384	86/160
Median	22/186	89/384	68/160
Erosion	30/186	60/384	68/160
2D Wavelet	40/186	83/384	84/160
IIR filter	28/186	52/384	44/160

We have also simulated the complete DSP C6201 CPU VHDL model which uses 14361 slices of the FPGA VirtexII that corresponds to 134% of available resources on the FPGA VirtexII-2000. Table 6 presents the slice used ratio and the percentage of corresponding modular VHDL DSP for each algorithm in comparison with a complete VHDL DSP C6201. These percentages have been calculated with numbers of slices used. On the average, these common image processing algorithms use only 30% of available resources of the DSP C6201. This allows us to implement several modular VHDL DSPs on a single FPGA in order to improve real-time performances. Table 7 gives obtained performances for each image processing algorithm using the multimedia board : number of modular VHDL DSPs realized, slices used ratio and the processing speed. These results show that we use available resources on the FPGA to a maximum, in an optimal manner, in order to increase real-time performances.

Table 6: *Results of the percentage of VLIW DSP for some common image processing algorithms. Percentages have been measured with hardware resources used on the FPGA.*

Algorithms	Slices used ratio	Percentage of VLIW DSP
Conv.	44%	33%
Median	32%	23.8%
Erosion	31%	23.1%
2D Wavelet	43%	32.1%
IIR filter	24%	17.9%

Table 7: *Obtained performances using the multimedia board: processing speeds have been measured with the image size of* 512×512 *pixels.*

Algorithms	Nb. DSPs	Slices used ratio	Processing speed
Conv.	2	88%	6 ms
Median	3	96%	8 ms
Erosion	3	93%	0.14 ms
2D Wavelet	2	86%	10 ms
IIR filter	4	96%	3.2 ms

5. CONCLUSIONS AND PERSPECTIVES

In this article, we present research results of enabling the DSP TMS320 C6201 CPU for real-time image processing applications, by exploiting FPGA technology. The proposed design method is illustrated and validated. Our experimental results demonstrate that modular VLIW processors specific to a target application can be implemented on an FPGA device in a very short design cycle.

Our approach applies some criteria for design tools: flexibility, modularity, performance, and reusability. In this paper, the target VLIW processor is DSP TMS320C6x. Our proposed design cycle can be generalized to other VLIW processors. An image processing algorithm can be as easily realized on an FPGA device as on a VLIW processor by a non-expert in electronics using the proposed method. This illustrates the usefulness of our approach for rapid prototyping of embedded systems. We have introduced modular Software/Hardware models. Only the bare minimum number of instruction sets necessary for a target application is implemented on DSP VHDL models. Thus, we can economize hardware resources of FPGA so as to implement several modular VLIW processors using a single FPGA.

In perspective, we would program an interface between the Mapping Algorithm Architecture help tool SynDEx and our VHDL DSP generator in order to facilitate the multi-processors development. SynDEx performs an optimized implementation of a given algorithm onto a given parallel architecture which allows us to improve memory models and communication protocols. For some complex applications, the VHDL code size of the generated C6201 CPU model will be too large. We can decompose these algorithms into several subtasks. We are considering the two options. First, we will implement some hardware components that perform specific tasks in order to increase real-time performance. Second, we also are considering an algorithm of several stages using capabilities of partial and dynamical configuration of VirtexII. SW/HW VHDL models for the next stage will be loaded in the FPGA at the same time as the current stage execution. For each stage, the generated SW/HW models correspond only to the minimum number of instruction sets of this stage.

REFERENCES

[1] K. Balakrishnan and N.A. Touba, *Matrix-based software test data decompression for system on a chip*, Journal of Systems architecture, Vol.50, pp.247-256, 2004.

[2] D. Gizopoulos, *Low-cost, on-line self-testing of processor core based on embedded software routines*, Microelectronics Journal, No.35, pp.443-449, 2004.

[3] M.A. Aguirre, J.N. Tombs et *al.*, *Microprocessor and FPGA interfaces for in-system co-debugging in field programmable hybrid systems*, Microprocessors and Microsystems, Vol.29, Issues 2-3, pp.75-85, 2005.

[4] M. Rupp, A. Burg and E. Beck, *Rapid prototyping for wireless designs: the five-ones approache*, Signal Processing, Vol.83, pp.1427-1444, 2003.

[5] P. Faraboschi, *The design of a technology platform for custom VLIW embedded processors*, Computer Physics Communication, Vol.139, pp.104-108, 2001.

[6] *TMS320C6000 Technical Brief SPRU197D (02/99)*.

[7] *http://www.xilinx.com*.