RECONFIGURABLE COPROCESSOR FOR MEDIA STREAMING

Sebastien Bilavarn

Signal Processing Institute
Swiss Federal Institute of Technology (EPFL)
Lausanne, Switzerland

Eric Debes

Architecture Research Labs
Intel Corporation
Santa Clara, CA

ABSTRACT

The development of more processing demanding video standards on one hand and the popularity of mobile devices such as digital cameras or wireless videophones on the other hand introduce a need of optimization at the processor level. Reconfigurable systems provides an interesting answer to this problem and several works have explored the possibility of performance and power optimization. The following study focusses on tuning a reconfigurable hardware to the requirements of future media processing, using DSP *operators appeared in recent FPGA families as an alternative to the typical ALU based architectures. In this paper, architecture perspectives are proposed with respect to low cost development constraints, backward compatibility, easy coprocessor usage and power / performance enhancement, using a new scalable data representation optimized for Quality of Services (Matching Pursuit 3D algorithms).*

1. INTRODUCTION

With the joint development of networking and digital coding technologies, multimedia and more particularly video services are becoming one of the major consumers of the new information networks. Regarding the expected evolution, the resulting infrastructure will be soon overloaded, so there is an obvious need of optimization. On the media representation point of view, a scalable system is needed in order to cope with different end user settings in terms of available bandwidth and processing power. High compression rates are also required to reduce the network load. The unavoidable counterpart of this is the computational complexity: if we consider the example of Matching Pursuit 3D algorithms (*MP3D*) that are currently under research at *EPFL*, the complexity of the decoder in its current version is about 100x more than a Discrete Wavelett Transform. This shows how processor execution is essential to achieve an acceptable Quality of Service (*QoS*) in video broadcasting.

Many works have addressed the problem of architecture tuning for multimedia processing, but our approach differs from those works as we focus on minimizing the modifications to bring to an existing CPU core using a reconfigurable coprocessor. The benefits of the approach are multiple: it can be applied to many existing microarchitectures, it keeps backward compatibility with previous algorithms, the design effort is low compared to a new microarchitecture design, so is the design cost and time to market. To achieve this, the following study is based on the use of a run time reconfigurable coprocessor and uses a prototyping environement combining an embedded CPU core (Nios processor) with a mixed grain reconfigurable hardware (Stratix FPGA). The definition of a working prototype will thus allow to check precise values about the effective speedups and power savings. This study considers a new scalable visual data representation based on (*MP*) paradigm as representative of future media processing constraints (video streaming in particular).

The outline of the paper is the following: first, an application-driven strategy based on Matching Pursuit algorithms is used to extract some key properties of the application domain and derive architecture impacts on the reconfigurable coprocessor. A prototyping step is then applied in order to check / refine the proposed issues and finally, first results and perspectives of future research are presented.

2. APPLICATION DOMAIN STUDY

MP is a new data representation currently under research at *EPFL* based on a progressive coding structure especially designed to cope with media delivery constraints. Basically, it decomposes a video signal into a stream of basis functions called atoms with their respective coefficients. Each atom can be considered as a spatio-temporal function generated by parameterized equations, so it can be represented by a limited set of parameters called indexes (7 parameters for 3D Matching Pursuit). This way of coding video introduces very interesting properties compared to other standards. It achieves very high compression rates as only the atom indexes and coefficients have to be transmitted and it offers high flexibility for multiresolution streaming or adaptive decoding [1]. Such interesting features has a counterpart and the price to pay is the computational complexity. This is mainly due to the atom generation process which is based on complex arithmetic computations (70% of the decoding time) but also to the atom representation based on large multidimensional arrays. Each atom is equivalent to a basis frame ($atom[x][y][t]$) which is generated from two mother functions: a 2D Gaussian function

$$g_1(x,y) = \frac{1}{\sqrt{\pi}} e^{-(x^2+y^2)} \quad (1)$$

and its 2^{nd} partial derivative

$$g_2(x,y) = \frac{2}{\sqrt{3\pi}} (4x^2 - 2) e^{-(x^2+y^2)} \quad (2)$$

by shifting, orienting, and scaling them. This processing is characterized by complex 32-bit fixed point computations repeated over each element of the array representation $atom[x][y][t]$. Video reconstruction is performed simply using a multiply-accumulate scheme (15% of decoding time) which is a well-known *DSP* primitive. *FFT* and inverse *FFT* (21 % of encoding time) are also used to reduce the complexity of the encoding algorithm.

MP computing characteristics join and exceed many previous results in the field of multimedia (*MM* in the following) computing characterization. *MM* applications are characterized by large amounts of data parallelism and high memory bandwidth [2]. This is especially true in the case of *MP* algorithms because of the atom representation based on large 3D arrays, with repetitive accesses to contiguous memory locations. Concerning this, the majority of memory bandwidth required for *MM* algorithms is known to be performed with a segment-oriented access pattern (e.g. filters, transforms [3]). *MM* computing is also based on native data types (8-bit pixels, 16-bit audio samples [4]) and 16 / 32-bit bit arithmetic computations. Such computations are often related to *DSP* primitives that exhibits patterns of computation based on multiply - add - sub and accumulate operations [5]. They represent a major bottleneck in *MP* processing because of the numerous 32-bit fixed point computations relative to the generating equations (1) and (2). The following section presents the impacts of such properties on the reconfigurable architecture.

3. ARCHITECTURAL ISSUES

3.1. First issues

Some prior works have already addressed a reconfigurable hardware tightly coupled with a CPU core. In [6] for example, a gate level reconfigurable coprocessor is built from off-the-shelf field-programmable gate arrays (FPGAs) to accelerate loops of general-purpose programs. Such systems give orders of magnitude in performance improvement but 1) compilation requires hardware design expertise beyond the knowledge of most programmers and 2) FPGAs perform gate level reconfiguration that leads to important reconfiguration times and energy waste. The search of an optimal tradeoff between efficiency and flexibility has led to an increase of reconfiguration granularity. As a result, decreasing the number of transistors used for device configuration reduces the energy waste, as some works have shown that 65 % of the power consumption was associated with the programmable interconnect in a FPGA [7]. This has also led to a significant reduction of the configuration delay and thus to enable system reconfiguration several times during execution. This property is referred to as "run-time" or "dynamic" configuration in Adaptive Computing Systems. An example of such an architecture in the field of mobile multimedia computing is given in [8]. In this system, a *reconfigurable datapath* has been introduced where the modules have various functions like multiplication and addition. This kind of solution is based on the exploitation of parallel datastreams using ALUs, in the manner of some vector architectures such as [2]. In the following, we address an alternative solution based on the use of dedicated execution units to simplify even more the design (controller in particular), enable coprocessor standalone execution and true concurrency with the CPU, and achieve better power efficiency. Given this, development costs, design time, backward compatibilty and time to market are important aspects of the design constraints, especially for mass market devices. Keeping the original microarchitecture and toolset development is possible considering a reconfigurable hardware tightly coupled with the CPU core and simply exploited by an extension of the instruction set.

Recently, FPGA families have integrated specific resources (mainly *DSP* operators and embedded memories) that can potentially accelerate *DSP* computing until several orders of magnitude. We propose to explore the efficiency of such concepts to optimize *GPP* architectures for this kind of processing. The coprocessor template we consider in a first approach resembles a typical FPGA architecture: it is based on a reconfigurable array of processing elements (*PEs*) associated with embedded memories and connected to a programmable interconnect network. To be efficiently interfaced with processor's data representation, we consider *PEs* are composed of 32-bit functional units and provide fonctionality for multiplication operations, accumulations, addition and subtraction, and summation, all of which are common arithmetic operations in DSP functions. Moreover, setting the granularity level to 32-bit allows to reduce the interconnection network complexity compared to a *FPGA*, thus achieving lowest reconfiguration times and fewer energy waste. Using dedicated functional units (multipliers in particular) is also expected to bring more performance and energy efficiency.

3.2. Coprocessor issues

Our initial objective was to execute full *DSP* primitives or computational intensive kernels on the reconfigurable hardware. But the complexity of the interconnect is an important factor that impacts greatly the system overall efficiency. Implementing full *DSP* functionalities requires to keep a great amount of configuration capabilities that leads to a high interconnection complexity. This implies complex reconfiguration schemes using configuration caches [6] or region partionning approaches which implementation may require important modifications of the basis microarchitecture of the *GPP*. Moreover, defining instructions specific to a given processing (*MP* atom construction for instance) present a low reuse potential. For a large programmable usage, a better solution is the definition of instructions with a high reuse potential so that they allow to maximize coprocessor utilization and thus the achievable speed up. Addind new instructions allows to keep compatibility with previous algorithms.

Considering this and previous domain study, we limit the use of the coprocessor to the execution of short sequences of operations. The high data parallelism can be efficiently exploited by processing several datastreams in parallel. In these conditions, the matrix structure is set to 8 columns of 3 Processing Elements (*PEs*), each allowing to chain a sequence of 3 operations max, which have been shown to be the optimal number of chained operations in the signal processing domain [5]. A *PE* includes one multiply and an add / subtract / accumulate unit associated with shift registers to implement multiplication / addition / subtraction / accumulation based patterns. 32-bit functional units are used to be compatible with the *GPP* internal representation.

A Direct Memory Access (*DMA*) module is added to allow for efficient bulk data transfers. It allows concurrent execution of the coprocessor with the *GPP* for

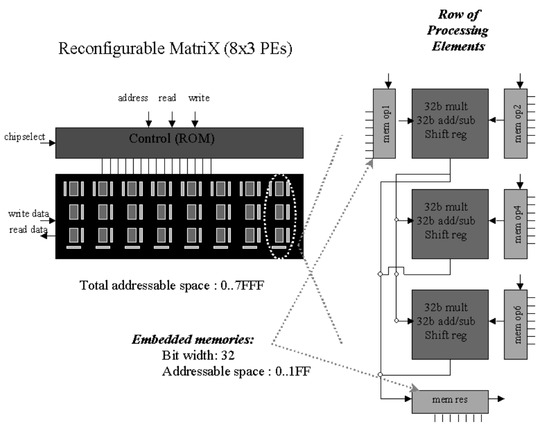


Fig. 1. Application domain impacts on microarchitecture

even better efficiency. Embedded memory blocks are used to feed *PEs* efficiently with their respective operand sets. Two operand memories are used for the first *PE* of a row, one for the second *PE* as one of the operands results from the first *PE*, and two for the last *PE* including result storage. Each memory is made accessible through its addressable space and always remains connected to the same *PE*, except the result memory that may be connected to the first, second or third *PE*, depending on the number of chained operations. Thanks to this, we provide standalone capability to the coprocessor and get rid of memory configuration problems. A reduced set of predefined and simple paths are defined in order to allow the chaining of 1, 2 or 3 *PEs* (figure 1). With such a simplified interconnect, reconfiguration can be achieved within a single clock cycle.

A similar optimization scheme can be applied for coprocessor control. As we ought to use the coprocessor for concurrent execution of regular processing, relatively few control is needed. A microcode ROM containing the configuration required in case of 1, 2 or 3 chained instructions is used and control is parameterized by the number of data processed to implement a fast configurable controller. The resulting architecture is given in figure 1.

4. RESULTS

The environement provided allows to compose custom bus-based systems out of library components such as CPUs, memory interfaces and peripherals, including user-defined peripherals. With the help of this, we designed the previously described 32-bit reconfigurable coprocessor and added it to a typical embedded configuration composed of a 32-bit CPU core with 16 KBytes data and instruction cache, associated with an external memory. Architecture development is still in progress and first results on the achievable speedup are exposed in the following.

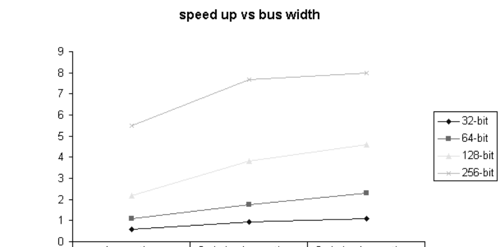
To compare the benefits of a coprocessor execution, we have first tried to compare a full hardware implementation with a full software implementation. To achieve this, we checked the speedup achieved when executing a basic code sequence (chained multiplication pattern) on the coprocessor compared to the same code sequence executed on the CPU core. From this point of view the obtained values represent the maximum possible acceleration, corresponding to a 100% coprocessor use (no software execution in this case). Results are presented in figure 2. We first implement the coprocessor with the original 32-bit system bus that limits strongly the achievable speedup to maximum value of 1.1. Unfortunately, the prototyping environment does not allow to modify the bus width, so we simulated a high bandwidth bus instead by writing the same data sets to each *PE* operand memory. Several speedup vs bus width measures have been reported this way and exhibit a maximum speedup of 8 for a 256-bit bus width.

We also tried to check the speed up benefits on a complete application with both software and hardware execution. The application considered is a 2 dimension Discrete Wavelett Transform optimized with a 3168 points add / multiply / add instruction and a 3168 points multiply instruction both intensively used in a lifting scheme process. The effective speedup on this example reaches a value of 3.6, the corresponding coprocessor use is 50 % of the total execution time.

Further development is currently in progress to reduce the data transfer cost and to extend the speedup measurements to other typical benchmarks, including power estimation.

5. CONCLUSION

In this study of architectural enhancements for future multimedia processing, we proposed to explore the relevance of a reconfigurable array of *DSP* operators to achieve high power and performance efficiency at low design costs. We defined a template architecture of a reconfigurable coprocessor that has been derived from the analysis of algorithms representative of future media delivery constraints, and previous well-known properties of *MM* computing. A FPGA prototyping board have been used to check the completeness and correctness of the proposed solution and first results



	1 operation	2 chained operations	3 chained operations
32-bit	0.58	0.92	1.1
64-bit	1.1	1.77	2.29
128-bit	2.2	3.83	4.6
256-bit	5.5	7.67	8

Fig. 2. Speedup vs bus width for 1, 2 and 3 chained operations

have shown performance exceeding software execution consistently until a factor of 8.

Future research will focus on the definition of a configurable memory structure in order to allow 1) random memory access with reduced bandwidth (for general purpose programs) and 2) local predetermined and segment-oriented access pattern [3][2] (to allow parallel access from all processing resources). The second aspect of the research perspectives concerns power consumption measures (directly on the prototype or using Altera power estimators) in order to check the energy savings expected from the sensitive execution time reduction.

6. REFERENCES

[1] P. Vandergheynst and P. Frossard, *Efficient image representation by anisotropic refinement in matching pursuit*, IEEE ICASSP, Salt Lake City UT, May 2001, vol. 3.

[2] T. Nguyen, A. Zakhor and K. Yelick, *Performance Analysis of an H.263 Video Encoder for Viram*, Int. Conf. on Image Processing (ICIP), Vancouver, B.C., Sept 2000.

[3] W. Hinrichs & al, *A 1.3 GOPS Parallel DSP for High Performance Image Processing Applications*, IEEE Journal of Solid-State Circuits, vol. 35, No 7, July 2000, pp. 946-952.

[4] A. Peleg and U. Weiser, *MMX Technology Extension to the Intel Architecture*, IEEE Micro, vol 16, No 4, 1996, pp. 42-50.

[5] M. Arnold and H. Corporaal, *Designing Domain-Specific Processors*, in Proceedings of the Ninth Int. Symposium on Hardware/Software Codesign, 2001.

[6] T. J. Callahan, J. R. Hauser, J. Wawrzynek, *The GARP Architecture and C Compiler*, IEEE Computer 33 (4): 62-69 (2000).

[7] J. A. J. Leijten, *Real-time constrained reconfigurable communication between embedded processors*, Ph. D. thesis, Eindhoven University of technology, November 1998.

[8] *http://chameleon.ctit.utwente.nl/*