Processor Enhancements for Media Streaming Applications

S. Bilavarn (sebastien.bilavarn@epfl.ch)
Signal Processing Institute, Swiss Federal Institute of Technology (EPFL)

E. Debes (eric.debes@intel.com)
Architecture Research Lab, Intel

P. Vandergheynst (pierre.vandergheynst@epfl.ch)
Signal Processing Institute, Swiss Federal Institute of Technology (EPFL)

J. P. Diguet (jean-philippe.diguet@univ-ubs.fr)
Lab. of Electronics and REal Time Systems, LESTER, University of South Britanny

Abstract. The development of more processing demanding applications on the Internet (video broadcasting) on one hand and the popularity of recent devices at the user level (digital cameras, wireless videophones, ...) on the other hand introduce challenges at several levels. Today, such devices present processing capabilities and bandwidth settings that are inefficient to manage scalable QoS requirements in a typical media delivery framework. In this paper, we present an impact study of such a scalable data representation optimized for QoS (Matching Pursuit 3D algorithms) on processor architectures to achieve the best performance and power efficiency. A review of state of the art techniques for processor architecture enhancement let us expect promising opportunities from the latest developments in the reconfigurable computing research field. We present here the first design steps of an efficient reconfigurable coprocessor especially designed to cope with future video delivery and multimedia processing requirements. Architecture perspectives are proposed with respect to low development cost constraints, backward compatibilty and easy coprocessor usage using an original strategy based on a hardware / software codesign methodology.

Keywords: video coding, matching pursuit, multimedia processing, reconfigurable coprocessor, software profiling, hardware design space exploration, codesign

1. Introduction

With the joint evolution of networking and digital coding technologies, video streaming on the internet is likely to be one the major consumer of the new information networks. The recent popularity of portable and handheld devices provides heterogeneity in terms of decoder settings (processing power, bandwidth) that have to be balanced first by an efficient visual data representation, and second by suited processor capabilities. To cope with this heterogeneity, future standards will have to be scalable in order to deliver several Quality of Services (QoS) within the same video stream. The unavoidable counterpart of this is to enhance processor execution in order to achieve the best video quality, real time decoding and power efficiency. In this paper, we address such an optimization study using representative algorithms in the field of video coding (Matching Pursuit 3D algorithms currently under research at EPFL) well-known to be a computationally demanding challenge in multimedia processing, and derive architectural issues on general purpose microprocessors. Until now, instruction set extension and dedicated hardware have been the most popular approaches to enhance a microprocessor execution for a given application or class of applications. Today, the maturity reached by reconfigurable technologies allows to explore new promising possibilities. Basically, a reconfigurable hardware unit can be efficiently used to execute many computational intensive kernels while the remaining parts of the algorithm executes on the general purpose core. In the following, we explore the relevance of a reconfigurable hardware solution to enhance future multimedia processing at low design costs.

The outline of the paper is the following: first, a review of state of the art techniques in the field of processor optimization is presented. Then we present an original method for tuning the original GPP architecture towards the application domain using a reconfigurable coprocessor solution. The strategy, which is based on the study of representative algorithms in the intended domain, will lead to derive some architecture impacts and define a first architecture template. Finally, future developments are presented.

2. Processor architecture enhancements

2.1. Optimization techniques

Given the algorithmic complexity evolution expected, the design of efficient architectures will be a key challenge for the near future. As an illustration, the complexity of a Matching Pursuit decoder, which is a new standard for scalable video coding, is about 100x higher than a Discrete Wavelet Transform for the same resolution level. The combination of this and the growing number of wireless and portable devices with low processing capabilities introduce a need for performance and power optimization. A lot of research is being conducted in this field; we present here a survey of the different trends, illustrated by some examples in the field of digital signal processing.

There are several ways to optimize programmable processors. Typical optimization schemes are based on specializing / adapting a given architecture to achieve better processing efficiency. Most of the techniques used rely on some kind of parallelization technique that can take place at several levels, from the instruction to the task level. The most popular approach addresses Instruction Level Parallelism (*ILP*): the architecture is tuned to the considered application domain by introducing specialized execution units and the instructions associated. Instruction set extensions consists then in detecting clusters of operations (sequences of instructions / recurrent patterns) that will lead to the definition of function specific blocks. Those specific units are then added to the datapath of a CPU and, usually, exploited using *VLIW* (Very Long Instruction Word) instructions. Several simultaneous operations are packed this way into a single *VLIW* instruction resulting in sensitive execution speedups. The extensive use of this technique have shown it outperforms general purpose processors both in cost, performance and power consumption. An alternative technique exploits data level parallelism with a well-known application to PC microprocessor being the Intel MMX technology [1]. The *SIMD* (Single Instruction Multiple Data) technique enables one instruction to perform the same operation on multiple data elements in parallel: 8 simultaneous 8-bits operations, 4 simultaneous 16-bits operations or 2 simultaneous 32-bits operations, for MMX Instructions. Thus it exploits the data parallelism inherent to multimedia processing and produces full application performance of 1.5 to 2 times faster than the same applications run on the same processor without MMX.

However, such techniques may not always be sufficient and further parallelization may be needed in order to meet higher power / performance constraints. Enabling concurrency at higher levels (functional / task) requires the design of a multi-processor system. Today, System on Chip (*SoC*) design makes the integration of several processors on the same die feasible, reducing drastically the delay and power consumption costs due to data transfers. Traditional heterogeneous solutions are based on *GPP(s)* augmented with dedicated (co-)processor(s). For example, some works in the field of wireless applications have naturally explored the combination of a general purpose architecture with a Digital Signal Processor [2]. The aim of this type of architectures is to benefit from the high processing power / low power consumption on the *DSP* side to provide maximum performance without sacrificing battery power. Further efficiency may be achieved using dedicated hardware to perform the co-processing of specific tasks. This approach have been carried out in [3] for example, where a specific co-processor has been designed for rendering of *MPEG-4* compliant video scenes. Basically, dedicated hardware presents the most effective way to optimize the overall cost-performance trade-off and to reduce power consumption. But, the efficiency of a hardware solution is at the expense of flexibility, as it is necessary to develop a new *ASIC* to implement changes or upgrades. That's the reason why a programmable solution is preferred.

Recently, reconfigurable computing introduced a new issue to this problem, combining a *GPP* with reconfigurable hardware. First systems were naturally based on *FPGAs*, and some works have shown very interesting results. In [4] for example, a *VLIW* reconfigurable processor has been tested on signal processing algorithms and showed speedups from 4.3x to 13.5x and energy consumption reduction up to 92%. Other examples have shown interesting results, like the Garp architecture [5] that combines a MIPS processor with reconfigurable hardware to accelerate loops of general-purpose software applications. Although these results are promising, optimizations are still possible in particular concerning the interconnection overhead. The main argument is that *FPGAs* waste a lot of energy because of the programmable network: a power breakdown of the *FPGA* shows that 65% of the power consumption is associated with the interconnect [6]. The search of an optimal tradeoff between efficiency and flexibility has led to an increase of the architecture granularity that have been made possible by the evolution of SoC size. Decreasing the number of transistors used for device configuration reduces energy consumption and reconfiguration times [7]. Of course, this results in a loss of generality, but achieving low reconfiguration times has introduced a new interesting feature: the possibility of dynamic configuration (e.g. at run time) that have led to the emergence of a new research area focussing on Adaptive Computing Systems.

Adaptive computing systems introduce a new promising design alternative, and this technology provides several benefits. Adaptive systems offer the potential for realizing cost effective systems as well as providing adaptability to changing system requirements, at run time. For example, a convolution is mapped onto a combination of memory, multipliers and accumulators. The same modules can be used in a different configuration to compute for instance a Fast Fourier Transform (*FFT*). This way, reconfigurability could bring the flexibility necessary to handle a variety of multimedia services and standards or to operate in dynamic application and communication environments. Several works in this field have explored the relevance of a matrix of ALUs to provide parallel processing capabilities [8][9][10]. Those projects report the best accelerations potential on typical benchmarks, but the impact on the design effort and the resulting cost overhead is prohibitive in a perspective of high diffusion consumer electronics. A low cost development issue ensuring moreover backward compatibility is possible by minimizing the modifications of an existing general purpose architecture.

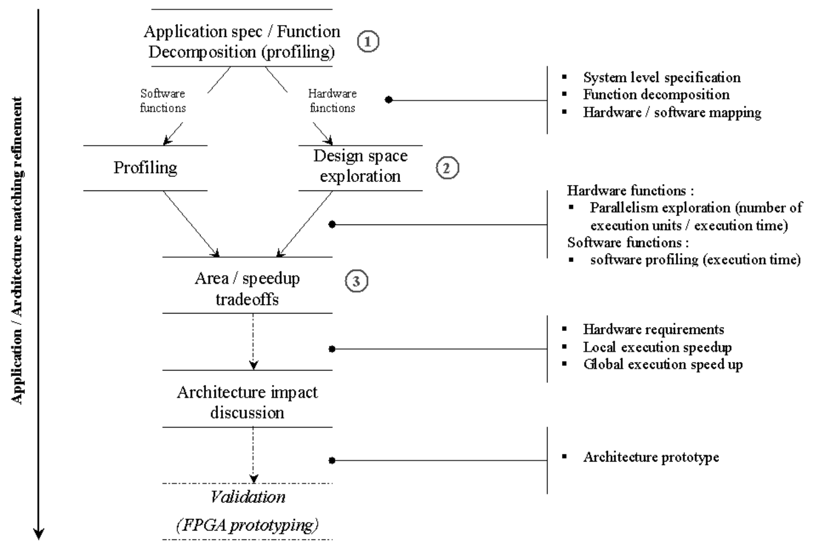

Figure 1. Design space exploration strategy : an application analysis at the system level based on a codesign methodology

2.2. Optimization strategy

As we target general purpose processors intended for consumer use, one of our main concern is to define a solution ensuring backward compatibility with previous existing architectures and algorithms. For that reason, we chose to explore the efficiency of a co-processor solution, as it does not introduce a lot of time overhead for microarchitecture design and tool set development. The co-processor simply connects the system bus and may be exploited by mean of an extension of the instruction set. The strategy used to design the reconfigurable hardware is derived from a hardware / software codesign methodology and is presented in figure 1.

A thorough application study is used to (1) identify critical processing functions in the application using a profiling step and (2) analyze parallelism i.e. several processing requirements (number, type and width of execution units) vs execution time tradeoffs. The first point allows to extract a set of critical functions (bottlenecks) that may be good candidates for hardware acceleration. For those hardware functions, a fast Design Space Exploration technique is used to highlight the acceleration potential for several processing requirements.

The methodology used for (2) is based on the Design Space Exploration methodology described in [11]. This method has been followed to guide the coprocessor design towards an optimal application / architecture matching. In particular, it allows very fast parallelism exploration from early specifications (C description) without the need of any time consuming hardware synthesis step. For a given specification, several area / performance tradeoffs related to different implementation opportunities (loop unrolling and folding, scheduling, binding) are evaluated. This parallelism exploration process does not consider the data transfer and reconfiguration times overhead, the timing values correspond to estimations that are computed using FPGA libraries. We assume here fast data transfers and low reconfiguration times, issues concerning those aspects will be tackled in a second phase. From this point of view, local speedups are derived to give a measure of the relative enhancement compared to a sequential (software) execution. Global speedups report the global algorithm execution enhancement when functions are mapped onto the reconfigurable hardware (3). The set of information obtained (speedup vs processing requirements) is used to discuss opportunities on the co-processor design / usage and derive some architectural impacts more suited to the design constraints and application domain requirements. The latter are analyzed in the following, with a special emphasis on scalable video coding using Matching Pursuit 3D algorithms, currently under development at EPFL.

3. Application study: Matching Pursuit

3.1. Matching Pursuit Algorithm Overview

MP3D coding [13] is based on the use of an approximation algorithm known as Matching Pursuit (*MP*). MP generates a naturally scalable decomposition of the video signal. This decomposition is then efficiently coded and transmitted to the clients. MP [14] tries to capture the spatio-temporal structures that are present in a sequence of 3-D space-time signal $I(x, y, t)$. It approximates the signal by a linear combination of generalized waveforms tuned to match the requested structures and selected among a vast library:

$$I = \sum_{i=0}^{N-1} c_i g_{\gamma_i}. \tag{1}$$

where g_γ is a function (i.e. an atom), and the set of all the atoms is referred to as a dictionary $\mathcal{D}$. MP provides an interesting generic solution to (1) by iteratively decomposing the signal using a greedy strategy. The set of atoms that composes the dictionary are tuned to best represent the video signal. The atoms are composed of separable 2D spatial functions and temporal functions. First, the spatial functions are spanned from two mother functions: a 2-D Gaussian function $g_1(x,y) = \frac{1}{\sqrt{\pi}}e^{-(x^2+y^2)}$ and its 2^{nd} partial derivative $g_2(x,y) = \frac{2}{\sqrt{3\pi}}(4x^2 - 2)e^{-(x^2+y^2)}$ by shifting, orienting, and scaling them. The function $g_1(x,y)$ represents the low-frequencies of the image whereas $g_2(x,y)$ deals with the discontinuities which are edges and contours. Second, the temporal functions are generated from β-spline $\beta^3(t)$ function by shifting and scaling it. β-spline is a smooth function that fits the motion model present in a video sequence. Those mother functions are used to generate the intermediate atom representations based on 3 dimensional arrays (atom[x][y][t]) used in the coding / decoding process. Compression performance is achieved through the set of coefficients c_i and indexes γ_i (7 parameters for each atom are used in video coding). The coding structure obtained allows progressive video reconstruction and quality when the number of atoms increases.

3.2. Decoder study

MP Decoding is an iterative process over the atoms performed in 3 steps: (1) Atom array construction from the indexes γ_i (function *ComputeNorm*), (2) atom normalization (function *SetPixelValue*) where all pixels are divided by the atom norm value and (3) video reconstruction (function *DecodeVideo*) performed using a simple MAC scheme. Profiling results obtained on a Pentium 3 mobile processor are presented in table I.

Functions *SetPixelValue* and *DecodeVideo* are relevant candidates for hardware acceleration because they are based on parallel divisions (that can be replaced by parallel multiplications in this case) and MAC patterns that are current computation schemes and can be reused for other DSP algorithms. For those two functions, hardware design space exploration using [11] have led to the results reported in figure 2. The processing requirements correspond to the number of execution units allocated, i.e. number of MACs for video reconstruction and number of dividers for atom normalization. The local speedup values show the important acceleration potential (up to a value of 430 using 256 MAC units for video reconstruction, and 330 for atom normalization using 256 divider units). Even though such configurations of 256 MACs / dividers are not realistic, lowest configurations still exhibit speedup potential of several orders of magnitude. To give an idea of the related enhancement on the full algorithm execution time, global speedups have been computed and does not exceed a value of 1.41. This can be

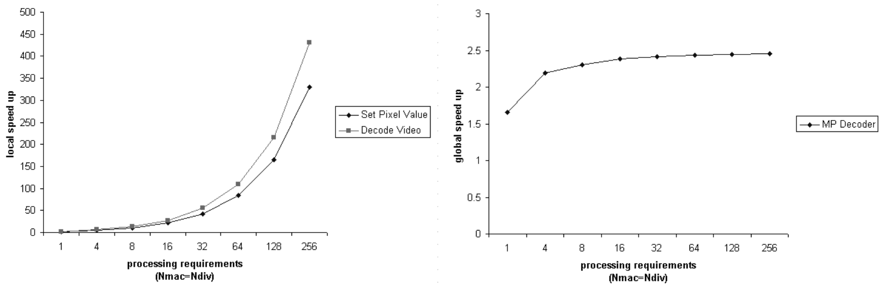

Figure 2. Decoder speed up estimation

explained by Amdahl's law [12] because of the remaining software functions executing on the GPP, especially the *ComputeNorm* bottleneck as it represents more than 70% of the total decoding time.

Table I. Profiling results of the decoder subfunctions on a Pentium III mobile processor

Function	Execution time	Distribution
Total	6.01 sec	–
ComputeNorm	4.25 sec	70.7%
SetPixelValue	0.76 sec	12.6%
DecodeVideo	0.99 sec	16.5%
Others	0,01 sec	0.2%

Consequently, one would like to speed up execution by executing this bottleneck (function *ComputeNorm*) on the coprocessor. But the computing complexity of this function (due to the generating functions of section 3.1) makes it difficult to implement the entire loop kernel without a strong impact on the architecture complexity, instruction design and reconfiguration times. Optimization with several instructions presenting low complexity and thus a higher reuse potential is a low cost solution: simple instructions are easy to configure more reusable, and allow to address the optimization of complex loop kernels. This shows the importance of defining a high reusable instruction set that maximizes the optimization potential, coprocessor use and thus the acceleration potential, as will be explained in section 4.

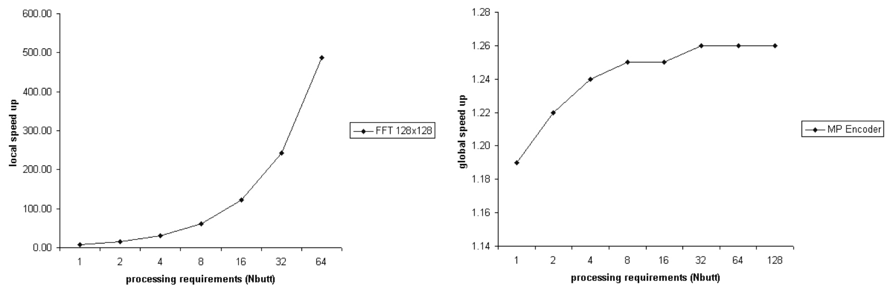

Figure 3. Encoder speed up estimation

3.3. Encoder study

Encoding consists in finding the best atom subset that will maximize video quality and compute the respective coefficients. For a given MP iteration, computing the scalar product of the current residual term with all possible atoms performs best atoms selection. To reduce the processing complexity, this computation is achieved in the frequency domain. FFT and Inverse FFT are intensively used (only the Inverse FFT represents 21% of the overall computations), so an efficient implementation is important for this application.

In order to analyze some execution benefits, we have considered several possibilities of hardware implementation of an FFT. Figure 3 reports the exploration results on a 128x128 FFT, and its impact on the global encoding time. The processing requirements correspond to the number of FFT butterflies (4 mult, 2 add, 2 sub) allocated. Like in the case of the decoder analysis, local speedups show a very important acceleration potential (up to 488) while the global acceleration remains below a value of 1.3 because of the software processing left unoptimized. Like in the case of the decoder, further enhancement is possible thanks to the definition of high reuse potential instructions (section 4).

4. Architecture impacts

4.1. Architecture Design Constraints

As stated previously, the considered architecture is based on a general purpose processor tightly coupled with a reconfigurable hardware. The coprocessor is dedicated to the acceleration of many computationally intensive tasks common in DSP processing using a coarse grain matrix of DSP ressources called Processing Elements (*PEs*) and connected to a programmable interconnect network. Given this, fast data transfers and low reconfiguration times are essential conditions to exploit the inherent parallelism potential of this structure. Issues concerning the processing requirements of PEs, memory bandwidth and reconfiguration to design a relevant and workable system reconfigurable at run time are addressed in the following.

Previous application study of MP algorithms have led to identify a set of critical functions for which a simple hardware implementation is possible. But on the other hand, the implementation of some complex functions or loop kernels creates two main issues at the system level. First, the complexity of a function impacts the ability to perform easy and fast reconfiguration. Complex mechanisms are required in this case like using partial reconfiguration support or configuration caches, or to perform dynamic task allocation and configuration loading. The impact on the design cost is high in this case. The second limitation concerns the development tools, in particular the definition of an efficient compiler considering reconfigurable hardware specificity (in particular, parallelism exploration). This is an issue that is currently addressed in the research community. Hardware configuration requires design expertise and complex synthesis procedures (high level and logic synthesis, placement & routing) beyond the knowledge of most programmers. The need of mature development tools limits strongly the scope of applications.

Fortunately, multimedia computing exhibits properties that may be efficiently used to define a simple usage without too much performance impact. The idea is to exploit the high data parallelism, high memory bandwidth and recurrent computation patterns requirements, well-known characteristics of multimedia computing, by restricting the coprocessor usage to short sequences of operations on parallel datastreams. As a consequence, we consider a row configuration of the matrix, where each row allows to chain a limited number of operations. This configuration enables the execution of several datastreams in parallel, each row corresponding to the processing of a datastream. The resulting simplicity achieves faster configuration, which is expected to be a few number of clock cycles. To address the compilation problem, we use a simple ILP approach and we extend the original instruction set of a GPP with high parallelism instructions based on common operations in DSP processing (like multiply / add / sub / accumulate used in FPGA DSP blocks). Thanks to this, the coprocessor can help the implementation of a wide class of DSP kernels such as the ones presented in table II. The resulting instruction set is based on the chaining of multiply, add, subtract and accumulate operations. To illustrate the

Table II. Class of algorithms suited for coprocessor implementation

Function	Specification equation
DCT	$X(k) = \alpha(n) \sum_{n=0}^{N-1} x(n).cos[\frac{\pi(2n+1)k}{2N}]$
FFT	$X(k, f) = \sum_{n=0}^{N-1} x(n).e^{-j2\pi kn/N}$
Filters	$X(k) = \sum b_n.x(k-n)$
Convolution	$(x * y)(k) = \sum_{n=-\infty}^{+\infty} x(n).y(k-n)$
Correlation	$(x \circ y)(k) = \sum_{n=-\infty}^{+\infty} x(n).y(k+n)$
Sqrt	$\sqrt{z} == \sum_{n=0}^{\infty}[(\frac{-1}{2})_n \frac{(1-z)^n)}{n!}]$
Log	$\log(z) = \sum_{n=1}^{\infty} \frac{(-1)^{k-1}(z-1)^k}{k}$
Exp	$e^z = e^{z_0} \sum_{n=0}^{\infty} \frac{(z-z_0)}{k!}$ at $z == z_0$

usage potential, a multiply accumulate instruction can be configured to speedup filtering and convolution computations, or an add multiply add instruction for a lifting scheme process in a Discrete Wavelett Transform. DCT and FFT execution can be optimized with butterfly patterns as they are based on multiply, add and subtract operations. Arithmetic functions can also be addressed like for instance a power function using several multiplications in cascade. The great number of possible instructions allows to maximize the coprocessor usage, thus the achievable speedup, but also to address the optimization of critical computation kernels executing on the software part. This is the case of the function *ComputeNorm* which is an important processing bottleneck in MP algorithms. The numerous arithmetic computations related to the generating equations (section 3.1) can be efficiently addressed using combinations of chained mul / add / sub operations.

4.2. Processing requirements

As stated previously, multimedia computing makes large use of multiplication operations, addition and subtraction, and summation, all of which are common arithmetic operations in DSP functions. The processing requirements of MP subfunctions (x axis of figures 2 and 3) exhibit such computation patterns at a 32-bit fixed point arithmetic granularity (e.g. MAC scheme for function DecodeVideo, FFT butterflies based on mul / add / sub operations). To cope with this constraint and to enable transparent interface with the processor data representation, we consider that PEs are composed of 32-bit execution units (figure 4). Setting the granularity level to 32-bit allows to reduce the complexity of the interconnect that has a great impact on recon-

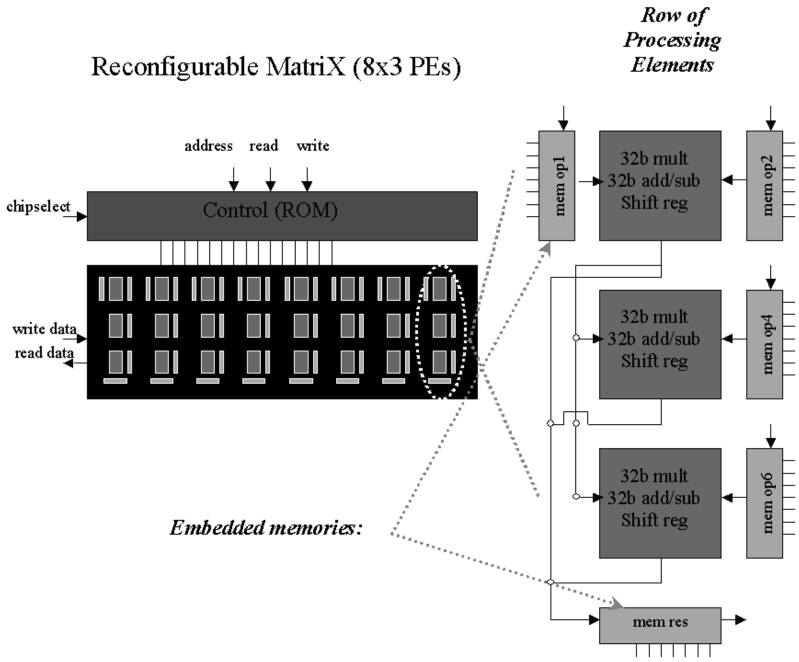

Figure 4. Processing Elements are configurable blocks based on 32-bit multiply and add / sub dedicated units. Embedded memories are used to enable stand alone processing on the coprocessor side. A predefined interconnect path is used to simplify the reconfiguration process.

figuration times and power consumption. As a result, a PE includes a dedicated multiply unit and an add / subtract / accumulate unit. The use of dedicated execution units (multipliers in particular) is expected to bring more performance and better energy efficiency.

Embedded memories are used to feed PEs with their respective operand sets and allows stand alone coprocessing. Given this, efficient schemes have to be used in order to reduce the data transfer overhead which is known to be a critical issue in multimedia processing.

4.3. Memory bandwidth

To enable concurrent processing of rows, an efficient scheme for simultaneous memory accesses is defined in order to keep the speedup benefits expected. A high bandwidth bus feeds several PEs in parallel with their operand sets. A 256-bit bus is used and allows to transfer

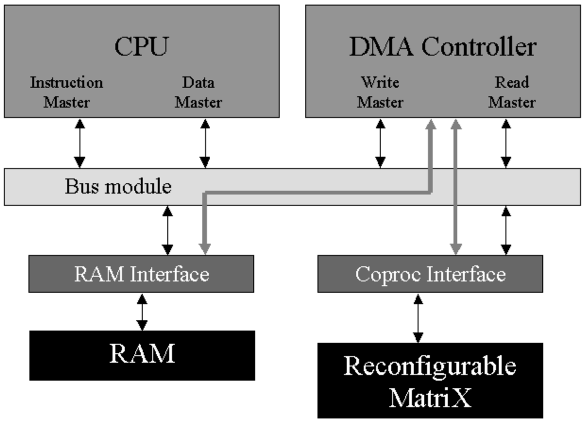

Figure 5. The coprocessor is based on a reconfigurable matrix of Processing Elements and embedded memories tightly coupled with the CPU. The DMA module allows fast data transfers and concurrent execution of the processors.

8 words simultaneously as we process 32-bit data. If we allow to chain a maximum of 3 operations, the resulting matrix structure is set to 8 rows of 3 PEs. As a result, the matrix may implement any combination pattern in a limit of 3 chained operations max, which have been shown to be the optimal number of chained operations in the digital signal processsing domain [15]. A pipeline schedule of operations can be applied to reach an optimal hardware acceleration.

With this 8x3 configuration, the achievable speedup can still reach values between 10 and 20 according to the timing estimations of figure 2. To address the data transfer overhead problem, a Direct Memory Access (*DMA*) module is added to allow for efficient bulk data transfers (figure 5). We remove this way the CPU from the datapath and enable true concurrent execution with the coprocessor for even better efficiency. Another DMA advantage is to provide time for the CPU to perform reconfiguration during data transfers.

4.4. Reconfiguration

A simple scheme is defined for memory configuration: each memory is made accessible through its addressable space and always remains connected to the same PE, except for the result memory that may be connected to the first, second or third PE, according to the number of chained operations in the current instruction to configure. This way, we get rid of memory configuration problems (how to connect dynamically to the bus ? dynamic memory size configuration ? address generation ?) and simplify the reconfiguration process. A reduced set of predefined path is defined for fast configuration of the interconnect. The predefined path is designed to enable the combination of any chained operations. This greatly simplifies the coprocessor usage, the reconfiguration and the instruction design: the coprocessor can be accessed like a memory R/W operation as each memory always keeps the same address space and processing is started when writing a specific address. The coprocessor configuration simply consists in parameterizing the instructions with the operation to chain (that may be contained in a configuration word) which configures the processing resources and number of data to process.

As we ought to use the coprocessor for regular processing (concurrrent sequences of a limited number of operations), relatively few control is needed. A generic sequencer for the pipeline schedule of 1, 2 or 3 operations is easy to derive as we do not need to configure the memories any more. The sequencer is then simply parameterized by the number of data to process (loop index) and the number of chained operations.

Given this, configuration of the matrix consists in writing the memories with their respective operand sets, configure PE operations, predifined paths and controller. The configuration steps are expected to be very fast and can be achieved during the DMA write operation. Then, operations are executed in the coprocessor and results are sent back to the processor memory. During that time, the CPU is fully available for software processing.

5. Conclusion / Future works

In this paper, we have presented an optimization study of general purpose processors for future media processing. A thorough application domain study has been performed using representative algorithms in the field of media delivery and considering typical DSP processing. A fast exploration and estimation methodology for the codesign of SoCs has been used to analyze the impacts of the hardware functions parallelism. Solutions have been proposed both for the coprocessor architecture and usage to cope with the application domain and hardware reconfiguration requirements to achieve an efficient run time reconfigurable system. With this system, speedups are expected to reach values between 10 and 20.

Future research will focus on the implementation of a working prototype. The environment chosen for prototyping combines an embedded processor core with reconfigurable hardware [16] that will allow to check the relevance of such an architecture and to report precise values about the achievable speedup and power savings.

References

1. A. Peleg and U. Weiser, *MMX Technology Extension to the Intel Architecture*, IEEE Micro, vol 16, No 4, 1996, pp. 42-50.
2. J.Chaoui, K. Cyr, J. P. Giacalone, S. Gregorio, Y. Masse, Y. Muthusamy, T. Spits, M. Budagavi, J. Webb, *OMAP : Enabling Multimedia Applications in Third Generation (3G) Wireless Terminals*, SWPA001, December 2000.
3. C. Heer, C. Miro, A. Lafage, M. Berekovic, G. Ghigo, T. Selinger, K. I. Wels, *Design and Architecture of the MPEG-4 video rendering Co-Processor TANGRAM*, ICECS 99, Cyprus.
4. A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Guerrieri, *A pipelined Configurable Gate Array for Embedded Processors*, FPGA '03, Feb. 2003, Monterey.
5. T. J. Callahan, J. R. Hauser, J. Wawrzynek, *The GARP Architecture and C Compiler*, IEEE Computer 33 (4): 62-69 (2000).
6. V. George, Hui Zhang, J. Rabaey, *The design of a low energy FPGA*, ISPLED 1999, pp 188-193, San Diego, 1999.
7. J. M. Rabaey, *Reconfigurable processing : The Solution to Low-Power Programmable DSP*, Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP '97), April 1997.
8. S. Keckler, D. Burger, *A Design Space Evaluation of Grid Processor Architectures*, 34th Annual International Symposium on Microarchitecture (MICRO), December 2001.
9. J. Helmschmidt, E. Schüler, P. Rao, S. Rossi, S. di Matteo, R. Bonitz, *Reconfigurable Signal Processing in Wireless Terminals*, DATE 2003: PACT XPP Technologies, Accent, ST Microelectronics.
10. *http://chameleon.ctit.utwente.nl/overview/main.html*
11. S. Bilavarn, *An Estimation and Exploration Methodology from System Level Specifications : Application to FPGAs*, FPGA 2003, Monterey.
12. G.T. Amdahl, *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*, Proc. AFIPS Conf., vol. 30, p.483, 1967.
13. A. Rahmoune, P. Vandergheynst and P. Frossard, *MP3D: Highly Scalable Video Coding Scheme Based on Matching Pursuit*, To be published IEEE ICASSP, 2004.
14. S. Mallat and Z. Zhang, *Matching pursuits with time-frequency dictionaries*, in IEEE Transactions on Signal Processing, vol.41, No 12, 1993, pp.3397-3415.
15. M. Arnold and H. Corporaal, *Designing Domain-Specific Processors*, in Proceedings of the Ninth International Symposium on Hardware/Software Codesign, 2001.
16. *http://www.altera.com/products/devkits/altera/kit-nios_1S40.html*