Exploring Processor and Memory Architectures for Multimedia

Iranpour, Ali

2012

Exploring Processor and Memory Architectures for Multimedia

Ali R. Iranpour

Lund 2012

LUND UNIVERSITY

This thesis is submitted to the Board of Research: FIME --- Physics, Informatics, Mathematics and Electrical Engineering --- at Faculty of Engineering, Lund University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering.

ISBN 978-91-976939-6-7
ISSN 1404-1219
Dissertation 38 ,2012
LU-CS-DISS:2012-1

Department of Computer Science
Lund University
P.O. Box 118
SE-221 00 Lund
Sweden

Abstract

Multimedia has become one of the cornerstones of our 21st century society and, when combined with mobility, has enabled a tremendous evolution of our society. However, joining these two concepts introduces many technical challenges. These range from having sufficient performance for handling multimedia content to having the battery stamina for acceptable mobile usage. When taking a projection of where we are heading, we see these issues becoming ever more challenging by increased mobility as well as advancements in multimedia content, such as introduction of stereoscopic 3D and augmented reality.

The increased performance needs for handling multimedia come not only from an ongoing step-up in resolution going from QVGA (320x240) to Full HD (1920x1080) a 27x increase in less than half a decade. On top of this, there is also codec evolution (MPEG-2 to H.264 AVC) that adds to the computational load increase. To meet these performance challenges there has been processing and memory architecture advances (SIMD, out-of-order superscalarity, multicore processing and heterogeneous multilevel memories) in the mobile domain, in conjunction with ever increasing operating frequencies (200MHz to 2GHz) and on-chip memory sizes (128KB to 2-3MB). At the same time there is an increase in requirements for mobility, placing higher demands on battery-powered systems despite the steady increase in battery capacity (500 to 2000mAh). This leaves negative net result in-terms of battery capacity versus performance advances.

In order to make optimal use of these architectural advances and to meet the power limitations in mobile systems, there is a need for taking an overall approach on how to best utilize these systems. The right trade-off between performance and power is crucial. On top of these constraints, the flexibility aspects of the system need to be addressed. All this makes it very important to reach the right architectural balance in the system.

The first goal for this thesis is to examine multimedia applications and propose a flexible solution that can meet the architectural requirements in a mobile system. Secondly, propose an automated methodology of optimally mapping multimedia data and instructions to a heterogeneous multilevel memory subsystem. The proposed methodology uses constraint programming for solving a multidimensional optimization problem.

Results from this work indicate that using today's most advanced mobile processor technology together with a multi-level heterogeneous on-chip memory subsystem can meet the performance requirements for handling multimedia. By utilizing the automated optimal memory mapping method presented in this thesis lower total power consumption can be achieved, whilst performance for multimedia applications is improved, by employing enhanced memory management. This is achieved through reduced external accesses and better reuse of memory objects. This automatic method shows high accuracy, up to 90%, for predicting multimedia memory accesses for a given architecture.

Contents

Preface

This thesis summarizes the results of my academic work in the Embedded Systems Design Laboratory (ESDlab) at the department of Computer Science, Lund University, for the Ph.D. degree in Computer Science. The main contributions of this thesis are derived from the following publications;

- *Evaluation of SIMD Architecture Enhancement In Embedded Processors for MPEG-4*, in Proc. Symposium on Digital Systems Design (DSD-04), Rennes, France, August 31 - September 3, 2004

- *Analysis of Embedded Processors for Streaming Media Applications*, in Proc. of the 8th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW-8), San Francisco, USA, February. 12, 2005.

- *Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors*, in Proc. Embedded Computer Systems: Architectures, MOdeling, and Simulation (SAMOS VI), Samos, Greece, July 17-20, 2006.

- *Performance Improvement for H 264 Video Encoding using ILP Embedded Processor,* in Proc. of the 9th Euromicro Conference on Digital System Design, Cavtat/Dubrovnik, Croatia, August 30th - September 1st, 2006.

- *Design Space Exploration for Optimal Memory Mapping of Data and Instructions in Multimedia Applications to Scratch-Pad Memories*, in Proc. of 7th ESTIMedia 2009, Grenoble, France, October 15-16, 2009.

Acknowledgements

Thanks to contributors goes here.

Lund, xxxx 2011

Ali R. Iranpour

Introduction

“Who Dares Wins”

Chapter 1

Introduction

In this chapter, we present a brief overview of the context and motivations behind the work presented in this thesis.

1.1 Background

Processor and memory architectures are fundamental components when exploring the design space in embedded and other resource limited systems. This resource limitation is in terms of power and energy as well as performance and cost, with the consequence of being a never ending trade-off between these factors. The ultimate goal is to get good enough performance in a reasonable power envelop. Taking a historic look at the evolution of a category of battery operated embedded systems, mobile terminals or handset, one can see an interesting transformation. These devices started as voice only with very limited computation performance, where at best they included a microcontroller, such as Zilog Z80 (8-bit microprocessor). This small processor was combined with other hardwired ASICs to handle the GSM communication protocol. These systems where often build on a printed wire board (PWB) where discrete ASICs, and other radio components where combined into a handset. The next generation of devices saw an integration of the digital discrete components to baseband ASICs and the radio components into RF ASICs. The digital basebands often included CPU, DSP and accelerators.

Around the time when second generations of mobile SoCs were introduced into handheld devices, the mobile CPUs were ARM7s and ARM9s, which became the dominant choice of processor, with their 32-bit RISC architecture and small caches. These processors provided enough processing power to not only handle the increased needs from the GSM data protocol, but also to handle the introduction of larger displays, multichannel audio, etc. The next big step was the introduction of many CPU/DSP systems with more memory, where each processor handled a predefined set of tasks and applications. These systems have further evolved into heterogeneous

multicore systems with symmetrical multi processing (SMP) cores, multiple DSPs, GPUs and multilevel heterogeneous memory system (caches and dedicated memory) where the entire systems is connected through a high bandwidth interconnect network-on-chip (NoC). Adding to this SoC system complexity and exponential increase in memory bandwidth needs the introduction of technologies, such as wide IO will create challenges in other areas, such as silicon technology maturity, manufacturing, standardization and overall costs.

Multimedia is an ever increasing focus area for all embedded systems on handheld devices. This multimedia centric view will drive development of multitude of applications that will manipulate multiple steams of real-time data, creating high computational demands resulting in increased system ramifications.

When combining the complexity of multimedia applications and resource limitations of embedded systems, the resulting system design provides some very interesting challenges, ranging from hardware and software partitioning to creating new system bottlenecks requiring handling of new trade-offs. Thus in order for achieving an optimal system design, a systematic exploration of the design space is needed.

In this thesis, we address design space exploration and analyze system requirements for handling multimedia applications in handheld embedded systems, such as advance smart-phones or PDAs. The first notion to take into account is “handheld” that basically means battery driven with all the constraints and limitations it poses on the system. The second notion is “embedded system” which means working in a resource limited environment. Sustained high computational performance cannot be achieved due to lower available power and energy budget compared to desktop system.

Audio and video processing are two major functionalities in multimedia applications today. They have many similarities, such as working on data streams with set of consecutive processing steps. Audio and video enhancements are the driving force behind many multimedia applications. Providing optimized system architecture to enable these processing needs will provide a well balanced multimedia embedded system.

Almost all multimedia algorithms work solely on fixed integer point arithmetic. At the same time video and audio differ in terms of computational complexity, in terms of algorithms and amount data being processed. Audio applications have come further in their evolution and maturity compared to video standards, where the algorithms used are often more computationally complex. The newer an application standard is, the less development time has been put in, making it less optimized, thus often requiring heavier computation and putting more demand on the memory architecture. Another aspect one needs to take into account is that applications not only evolve to more complex applications but can also move “back” to less complex, when introducing new features and other enhancements, such as post processing, etc. This increases the demand for the being able to handle and reuse generated data even further. Audio

applications have come further in their evolution, making it possible for having significantly more post processing enhancements compared to the video applications. In the case of video, many of the fundamental issues for handling data has not been fully solved, thus focusing on post processing is not where the main research effort has been up now.

As mentioned previously, in order to handle multimedia, there is a need for system design that is well balanced both in terms of power as well as performance. At the same time it gives the desired flexibility for being able to handle different algorithms and different multimedia applications. The two hardware components which are critical for achieving a well balanced embedded system are processing and memory demands. One needs the computational power to run the algorithms, but at same time the memory design needs to be able to feed the processing elements. The required processing differs significantly between audio and video applications. In some cases this can be a magnitude higher for video compared to audio, for example when comparing eAAC+ audio codec with H.264 video codec. The memory requirements differ both in terms of footprint but also in terms of bandwidth.

The power consumption is also a key factor along with performance for battery operated devices. When it comes to power consumption it is not only processing but also memory that needs to be considered. Often design decisions are made where the focus is on reducing the processing power consumption. This can be a costly approach as this is often paid for in increased memory traffic, potentially resulting in worse overall power consumption. Good designers need to take into account the entire system.

When it comes to processing and the potential speedups there are two approaches for multimedia applications. The first and most used for multimedia is data level parallelism (DLP). The other much less used in multimedia is instruction level parallelism (ILP). The straightforward approach would be to parallelize all potential available parallelism in the application by widening the data paths and increasing the number of processing elements. This however is not the most efficient usage of silicon nor is it power efficient, as not all parallel parts of the application are heavy contributors to the overall computational workload. The power consumption increase comes from over-usage of the memory system, interconnects and external memory accesses. Additionally, the complexity of multimedia algorithms could result in even worst overall performance as it is very easy to get penalized by wrong partitioning due to increased memory accesses.

There are a number of ways to get the DLP in multimedia. Single instruction multiple data (SIMD) is one common approach that can be utilized. This is often designed as instruction set extensions to vector instructions existing processor designs. Another approach is using digital signal processors (DSPs). Yet another approach is to increase the number of processor cores, such as in symmetrical multi-processors (SMPs). Depending on the multimedia application algorithm and data usage one approach maybe more suitable than another.

In the case of ILP, both very long instruction word (VLIW) and superscalar architectures are often used. The concept of superscalar design basically means by increasing the width of the processor and enabling multi instruction dispatch, and execution in every clock cycle thus increasing the overall throughput of the processor. In the case of VLIW architectures, multiple 32bits instructions are combined in to a single VLIW instruction which could be up to 128- or even 256-bits long.

Memory design for multimedia is very much connected to application algorithms and their behavior, as these factors directly control the access patterns and bandwidth requirements. The memory designs range from homogeneous designs with fully dedicated memory, to fully cache based design or heterogeneous memories, utilizing multilevel memory hierarchy. Depending on the chosen architecture, there are different trade-offs to be made, involving memory sizes, types of memories, number of levels in the hierarchy and number of memory ports. When it comes to type of memories there are different caches with different associativity and policies. There are also scratch pad (SPM) or tightly coupled memories (TCM), called in ARM architectures, and content addressable memories (CAM). The SPM is memory design often used in real time and multimedia applications due to its low latency and predictability, as SPM are managed by software. CAMs are on the other hand used when memory speeds are essential, for example in high-speed network switches, as CAMs can search the entire memory in a single operation. The CAMs work in "reverse", basically unlike standard memory, in which the address is given and then the content is returned. In CAMs the data is supplied and the memory returns the address if the data is in the memory.

This thesis answer the following questions: how can we in an optimal way map data and instructions to a particular architecture memory? This is combinatorial problem with discrete set of solutions, where we are looking for the best solution for a specific architecture given a set of memory objects. There are different techniques for solving this, either by using heuristics or complete methods, such as constraint based approaches.

This thesis also addresses another important question, the exploration of the processing and memory design for any given multimedia applications. The important aspect of the multimedia application when looking into system architecture using design space exploration is the abilities of the application itself. Basically evaluate the application in realistic fashion with real world needs so that simulations are as close to reality as it can be. This can be very difficult to achieve, given the real world complexities. So it is very important to use real state-of-the-art world multimedia application for design space exploration.

1.2 Motivation

Multimedia has become one of the most important factors in our day to day lives. It has been one of the main driving forces behind the last century's rapid evolution of human society. What started in the beginning of the last century as rudimentary moving

pictures and recorded music has evolved in to today's high definition multimedia content. On top of this evolution there has been as accelerated exponential grows in the last decades that involves the concept of mobility. A notion that enables anyone to use and generate multimedia contents any where at anytime.

The question to ask is what technical advancements have made this possible? What kind of technical trade-offs were made to get to this point? Also what are the hurdles and challenged that lie ahead? An "always connected" society is the major reason for this growing multimedia appetite, major reason being the mobility and flexibility that the devices offered. This led to a consumer appetite for non-stop multimedia content. The device performance, usability and durability were also a huge contributing factor for this growth. This phenomenon involves a wide variety of reasons from cultural to technical reasons, in this thesis we focus on how multimedia content is handled, with emphases on processing and memory for handheld battery powered device. The target is to be efficient in terms of power as well as performance with as flexible architectures as possible. This is a classical computer architecture trade-off problem where the task is to find the right balance when taking in all different factors into consideration.

In the case of processing, as stated earlier, multimedia applications have specific properties that exhibit various parallelisms. These are not only data level parallelism DLP but, depending on the specific algorithms, also instruction level parallelism ILP. In order to get the most out of DLP and ILP there are a number of techniques and architectures that can be utilized, such as SIMD, VLIW, SMP, superscalarity. The architecture we focus on in this work was SIMD together with superscalarity, as this provided both flexible architecture and gave enough performance in the right power envelop for handling multimedia.

As mentioned in the previous section, there are many options for memory architectures for multimedia. They range from straightforward cache based to more complex solutions with different dedicated memory buffers and combinations of the two. Keeping in mind that not only performance but flexibility of being able to execute different kind of applications, both multimedia and others, are intended to use the same memory system, a balanced memory system was necessary. The architecture choices and the complexity of multimedia applications raise the question, what memory architecture suites the multimedia scenario? This can be done with many different approaches and methods. In this thesis we propose a systematic approach to find an optimal mapping of data to heterogeneous memory architecture, consisting of a level1 SPM with multilevel cache hierarchy.

Looking at different approaches for optimization, and specifically optimal mapping of data, one of the best known is integer linear programming, were the problem is modeled using integer valued variables and linear inequalities. Another approach is heuristic based algorithms. The approach chosen in this case is based on constraint programming (CP), where by using constraint specific reasoning methods, a problem is formalized and a model of the problem is created. This is then used as input to a solver

that can find one or many solutions for the specified problem. The constraint solver used in this thesis was JaCoP [1]. The problem we are solving here can be broken down to a combinatorial optimizations knapsack problem. The work in this thesis is one of the first using constraint programming for this kind of problems. The main benefit when using CP compared to other method is that we can state the problem in a flexible way and obtain the optimal data mapping solution for specific memory architecture.

1.3 Structure of the thesis

The thesis comprises of two parts. The first part gives an overview of the area, background and motivation behind the work. The second part consists of collection of published papers. Chapter 2 presents the challenges in embedded system with multimedia. Chapter 3 discusses related work and in chapter 4 the published papers are briefly presented. Chapter 5 and 6 gives a list of contribution, conclusions and future trends.

Chapter 2

Challenges with Multimedia Applications in Embedded Systems

2.1 Embedded Architectures for Video and Audio

There are many challenges that lay ahead when designing today's multimedia embedded systems. These range from designing a system that needs to work in a diverse environment, where frequently many different types of tasks and applications need to be handled, to being at the same time designed with power limitations of battery operation. These challenges are often contradicting with constraints, such as performance and power consumption but also cost and time to market. Breaking down these constraints, certain design areas stand out, such as processing and memory.

The different architectural approaches for handling the processing and memory requirements depend on the selected applications, such as video, graphics and audio. In this thesis, we have focused on video and audio applications, more specifically on block based video codecs, MPEG-4, H.263 and H.264. For audio we have focused on compression codec eAAC+, which is a sub-band codec operating in the frequency domain.

An important factor when doing research on design space exploration in the embedded domain, especially for multimedia applications, is to use realistic applications representing industry state-of-the-art implementations. It cannot be emphasized enough. This is important both in terms of avoiding kernels for system evaluations but also avoiding sub-optimal code, which is often the case in reference applications provided by standards. In most cases design choices made, based on these applications or kernels, are not relevant or even erroneous. Often the focus lays in speeding up parts of applications or kernels, which at first glance seem too computationally intensive.

However in real applications these parts can been solved by more sophisticated algorithms with less computational intensity or even be avoided altogether.

The main architectural solution that has been proposed in embedded domain, targets the inherent parallelism present in multimedia applications. In terms of architectures for processing, these solutions range from using standalone dedicated hardware to using fully software based solutions based on general purpose processors (GPP). The memory solutions used in embedded domain for multimedia applications address mainly throughput issues that are caused by bandwidth limitations. The solutions utilize different on chip memory designs, such as hardware controlled buffers, different type of caches and scratch pad memories (SPM).

In this thesis, the aim has been to address multimedia (video and audio) applications for processing and memory addressing constraints that exist in embedded domain. This has also included a need to propose a method for optimal mapping of memory objects to a specific architecture.

2.2 Multimedia Processing

The digital workload for handheld devices, such as Smartphones, including all control, data and signal processing activities, is nearly 100 Giga operations per second (GOPS). This workload increases by an order of magnitude every 5 years [2]. Keeping in mind that all these functions have to run for hours on a single battery charge, and standard Li-ion battery has around 4.5Wh of energy (1200mAh at 3.7V), this is extremely challenging. This basically means we have a tight total power budget, not taking thermal issues into consideration. The cellular transmitter (GSM, UMTS or LTE) takes on average 1-2 W. The rest is what is available for the digital workloads. Comparing the CPU in a Smartphone with a PC, it has to run more than two orders of magnitude more power efficient, 0.2 W vs. 30W [2].

Looking at processing 100GOPS, the thought of using a single processor to handle this is out of a question as it implies running a processor at 100GHz! Looking at the other side of running 1000 cores at 100MHz is also unreasonable with all the overhead traffic and costs. The only way forward is using a heterogeneous architecture based on programmable cores and hardwired functional accelerators.

Multimedia applications are major contributor to the 100GOPS workload. These applications put huge demands in terms of processing on any system, especially on resource constrained embedded systems. Regardless whether these systems are hardwired or fully flexible, the entire system (computation units, buses, memories) is often pushed to peak performance edges. What kind of processing solution is chosen is often a trade-off between required performance, flexibility, available power budget and available technology.

All processing solutions address parallelism present in multimedia applications. This can be achieved through different approaches, such as dedicated hardware solutions, or

programmable solutions, where they can either be homogeneous architectures, such as multicore architectures, or heterogeneous systems, where programmable cores are combined with hardwired solutions. Further a heterogeneous system where dedicated hardwired solutions are present can either be loosely or tightly coupled. What is meant by loosely coupled is where, for example, a system is designed around a general purpose processor (GPP) with dedicated hardwired blocks for specific accelerations. A tightly couple system is a GPP enhanced with instruction set architecture (ISA) extensions, such as SIMD style extensions.

Dedicated hardware solutions for multimedia are often designed to perform single or very few specific tasks. They can range from performing parts of computationally heavy algorithms, to executing entire applications, such as encoding a specific H.264 bit stream. The key here lay in performing a predefined task with high performance but at expense of flexibility.

Using either general purpose processors (GPP) or digital signal processors (DSP) is a common approach for achieving flexibility with limited performance requirements. GPPs are designed traditionally to enable efficient execution flow and DSPs are designed to perform well for constant-rate data flow applications. Together, GPPs and DSPs can also be combined to processing engines for a wide variety of multimedia applications and products. These processor based solutions can be used for wide variety of applications, but their flexibility comes at a cost of being limited in terms of performance. For example, in the video encoding and decoding, when the resolution is above a certain level, such as high definition (HD), the performance is not enough.

An approach which can provide better performance compared to above solution is when GPPs are combined with dedicated hardware in a loosely coupled heterogeneous architecture. Here specific computationally heavy blocks are connected together with GPPs via buses or interconnects. This is a very simple and straightforward hardware approach but has limitations in terms of flexibility. This hardware simplicity comes at expense of software complexity, making programmability of these systems difficult and many times inefficient due to potential hardware software overhead costs and traffic.

In order to keep high flexibility but provide multimedia capability, microprocessor manufacturers have introduced tightly coupled heterogeneous extensions to their instruction set architectures (ISA) that enhance the performance of multimedia applications. These ISA extensions operate in a SIMD fashion to exploit data level parallelism (DLP) in multimedia applications. SIMD enabled microprocessors increase the processing capability while offering the low power consumption required by handheld, battery powered devices.

Multimedia applications typically operate on narrow words (primarily 8- and 16-bits) and spend a significant portion of execution time in loops that have a high degree of processing regularity. Packing several small data elements into a wider data-path

enables simultaneous processing of separate data elements. Such SIMD instructions are available for several arithmetic and logic operations in addition to special media operations, such as sum-of-absolute differences (SAD). SIMD capabilities enable more efficient software implementation of high-performance media applications such as audio and video codecs. Processors, such as ARM NEON, MIPS MDMX and Intel Architecture MMX and SSE, provide SIMD style instructions.

For example, ARM as one of the major embedded processor providers, introduced NEON [3], a combined 64- and 128-bits wide general purpose SIMD extension to the ARM instruction set architecture to accelerate multimedia applications, such as video and audio encoding/decoding. NEON is tightly coupled with the ARM core providing single instruction stream with unified view of memory. NEON supports 8-, 16-, 32- and 64-bit integer and single-precision (32-bit) floating-point data and operates with up to 16 operations at the same time. NEON is capable of delivering between 60-150% performance improvements for specific multimedia codecs [3].

Another closely related aspect to take into account regarding multimedia processing is that most research has been focusing on the data centric parts of the applications. This has resulted in neglecting of the potential parallelism available in the control parts. The importance of these parts is becoming even more critical as multimedia applications are evolving towards more advance control dominated applications [4]. This trend can, for example, be seen in the evolution of video codecs, MPEG2, MPEG4 and H.264. This potential parallelism can be exploited by instruction level parallelism (ILP) through the usage of different architectures, such as very long instruction word (VILW) and superscalarity.

As mentioned in the previous section a specific solution depends very much on the selected applications. More specifically the processing of video and audio has similarities but also many differences. Audio is a data dominated application making it easily parallelized, which is often performance enhanced by parallel architectures, such as DSPs, SIMD and parallel dedicated hardware. Video on the other hand is not as strait forward as audio in mapping. It has parts that are data dominated and have significant parallelism, such as block filtering and motion estimation. But when it comes to compression and decompression this is control dominated [5]. It is essential to understand which type, data or control dominated an application is.

Processing needs differ therefore significantly between audio and video. Audio applications have significant lower processing, almost a magnitude lower, less than 100 Mega cycles, compared to video processing. There are two main reasons for this, first the amount of data being processed is significantly lower for audio applications. Second audio applications work on one dimensional data arrays making audio applications much easier to process and predict. Video applications can be seen as two dimensional arrays.

In the subsequent sections we will discuss, in more details, video and audio processing.

2.2.1 Video processing

The main multimedia video trend in handheld, battery operated devices is an increased complexity of the video codecs, an increased number of supported video codecs and an increased resolution for both cameras as well as displays. This has led to a significant increase in processing demands.

Processing requirements for video applications are significantly higher than audio applications as the amount of data needed to be processed is higher. In addition, the compression efficiency of newer codecs, such as H.264/AVC, has significantly improved. This has come at a cost of computational complexity and memory access bandwidth. For example, for HD720p video (1280x720@30fps, ±128-pel search range, 2 ref frames) encoding the required off-chip memory bandwidth with level C scheme is 1071MByte/s [6]. Comparing the number of pixels for a standard definition (SD480p) to a full high definition (HD1080p) this about 6 times more pixels per frame for the HD (350000 vs. 20000000 pixels/frame). This equals to a processing need of over 3200 Mega-cycles/s for HD1080p and 550 Mega-cycles for SD480p for a H.264 video stream.

To understand better video codecs functionality and their processing requirements we will discuss H.264 codec in more details. The main blocks in H.264 encoder and decoder are illustrated in figure 1. As seen in figure 1 the entire H.264 decoder is a subset of the H.264 encoder. The major processing demanding blocks are motion compensation and motion estimation. Other parts in H.264 encoder include the selection between intra- and inter-coding for block-shaped regions of each picture. Intra-coding uses various spatial prediction modes to reduce spatial redundancy in the source signal for a single picture. Inter-coding (predictive or bi-predictive) uses motion vectors for block-based inter-prediction to reduce temporal redundancy among different pictures. Motion vectors and intra-prediction modes may be specified for a variety of block sizes in the picture. Finally, the motion vectors or intra-prediction modes are combined with the quantized transform coefficient information and encoded using entropy code such as context-adaptive variable length codes (CAVLC) or context adaptive binary arithmetic coding (CABAC).

Consider for example, parallel arithmetic instructions, such as sum of absolute difference (SAD), a multi-cycle operation that is often pipelined for accelerating motion estimation in video compression codecs. These instructions are used to perform parallel half-word or byte wise arithmetic operations. This is used in motion estimation, which is one of the most critical components in a video coding system, and it also dominates the major part of the computation complexity and memory bandwidth. Depending on implementation, in straightforward dedicated hardware implementation, motion estimation is an I/O bound problem rather than a computation bound one [7]. Looking at motion compensation the most computationally demanding parts are half-pixel interpolation, which is done by using a 6-tap FIR filter, and quarter-pixel interpolation is done by averaging two half-pixel values. These three operations, SAD,

half- and quarter-pixel interpolation can easily stand for up to 40% of the overall H.264 encoding time [8].

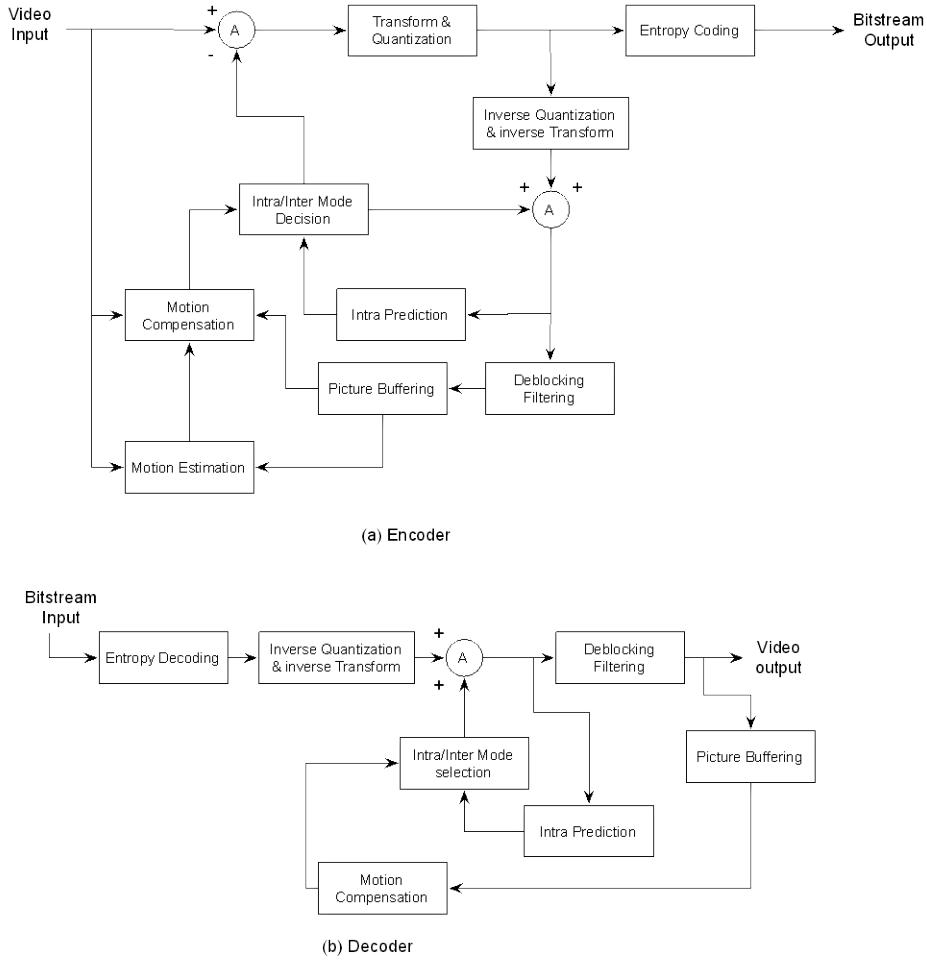


Figure 1. Block diagram of H.264 algorithm encoder and decoder.

When looking at video codecs, the encoders have more flexibility to do trade-offs with performance. The SADs for a motion estimation search in the encoder, can assign the motion search within each of a range of reference frame to a unique thread, thus increasing the possibility of parallel execution. Looking at video data, it is represented in most applications as 8 bit resolution. The size of parameters and coefficients are

coded in 16 or fewer bits. The filters coefficients and algorithms of H.264 and many other popular video codecs are chosen to ensure that even intermediate values of calculations will not overflow 16-bit arithmetic's operations within processors [7,9]. The adders, shifters, ALUs, multipliers and other data processing elements within programmable video processors are typically 16-bits wide, which will directly impact the system architectures.

Other important aspect of video processor is the pixel rate requirements of the application that must be met as it is directly connected to amount of data needed to be processed. The pixel rate is the product of height, width and rate at which full frames are displayed. For many video coding standards the worst case time difference is at least a magnitude greater than the typical sequences.

Another factor to bear in mind is the explosion in supporting higher and higher screen resolutions which has come at same time as an increase in codec complexity when comparing H.263 with H.264. An increase from H.263 at QCIF to H.264 at SD has a computational complexity increase of about 65 times [4]. This has resulted in huge changes in video system architecture. In the mobile domain, there is also an increased number of video codecs that need to be supported. Here is a short list of codecs that are mandatory, MPEG2, MPEG4, H.263, H.264, VC-1, RV, DivX, VP6/7, Sorenson Spark and AVS 1.0. With this increased number of codecs has come increased computational complexity as well, for example, comparing H.263 with H.264 on the same resolution and frame-rate H.264 is 2.5x more complex [4]. At the same time, the fast pace of mobile handheld device development has meant that there has always been a request for supporting multimedia codecs that were not available when the chips were designed. All these requirements lead to a need for a flexible video processing architecture with high performance.

The biggest trade-off in video hardware systems is flexibility/programmability versus dedicated hardware. This flexibility allows late adaptations and development of new multimedia applications, but it also gives the opportunity for increased differentiation by enabling new unique features and improvements to be added later. This increased flexibility comes often at price of increased power and area and in some cases of decreased performance. As mentioned in the previous section, the common denominator for all video processing solutions is exploiting the inherent parallelism that exists in multimedia applications and especially video applications. The main architectures range from fully dedicated hardware solutions, to the full programmable general purpose multicores. In between one can find more or less programmable solutions with specific hardwired architectures. These categories of architectures include heterogeneous uniprocessor solution with SIMD extensions, which we have focused on in the work presented in this thesis.

Fully dedicated hardware is designed to perform a predefined function, resulting in inflexibility for handling new video codecs with updated requirements. Programmable processors on the other hand provide flexibility but at increased power and lower

performance. The implications of this provided flexibility, is the possibility of performing complex functions by combining various simple functions in the order specified by a software program. This leads to many more reads and writes that are performed in a programmable processor, than in a dedicated hardware. Each read and write takes time, reduces overall throughput, and consumes power, wastes energy and generates excess heat. For motion estimation algorithm dedicated hardware solutions often adopt full search block matching algorithm (FSBMA) [10] because of predictability and regularity of the computations. This however requires maximum computation load and memory bandwidth. The reason for the large computational load is due to a need for working with a large set of candidate blocks that needs to be matched. Also the huge memory bandwidth results from loading the reference pixels for candidate blocks. Dedicated hardware logic yields best performance per cost, but keeping in mind the need for supporting multiple coding standards, handling new proprietary functions, and future changes to application requirements a programmable processor is preferred solution.

Programmable processors are preferred for applications that support many standards and in areas where new standards evolve in a fast pace, such as in video. A closely related factor that needs to be taken into account is porting an existing solution into new release of operating system. This is significantly easier for more flexible solutions.

Another aspect regarding fully programmable processor solutions is the requirement of having sufficient internal storage to efficiently perform its most demanding video processing functions. This consumes large silicon area, resulting in greater manufacturing costs.

A heterogeneous architecture (mix of hardwired and programmable core) achieves best balanced solution and is probably the most dominating solution for video processing. For achieving a robust and balanced architecture a carefully made partitioning between the hardwired and programmable cores needs to be made [4], as this is to reduce the hardware software interactions to minimum to achieve a good level of efficiency. Basically, the hardwired accelerators should be used for processing large chunks of data. This also relives the memory system as memory burst are optimal for external memory bandwidth. This results in architecture, where the only tasks left for the programmable part is control code. Looking at different design options for the programmable part, a general purpose CPU is much better choice compared to a DSP or a VLIW, as neither is optimal for handling control code [4]. Many newer video codecs, such as H.264, have different control flow in the algorithms, depending on if some of codec features, for example flexible macroblock ordering (FMO) etc. is used or not. Also post processing is another area which is very difficult to implement in hardwire, as algorithms for these tasks are very application specific and could often have complex control code thus requiring significant programmability.

Programmable video processor architectures achieve best performance through the use of parallelism at the data- , instruction- , and task-level. This is usually implemented

using SIMD, VLIW and multicore. On top of these an optimally sized ALU, multiplier, and load/store data paths is essential. In programmable video processors, video processing functions that are commonly accelerated with specialized hardwired logic include, multi-pixel sum of absolute differences (SAD) for motion estimation and filtering such as for inter-pixel interpolation (FIR) and deblocking filters. Where the video processing algorithm has data parallelism the performance of a processor is greater if the processor has single instruction multiple data (SIMD) instruction that, with the execution of a single instruction, operate on adjacent data elements in parallel. SIMD extensions have been added to almost all commercial general purpose processors table1.

Table 1. List of SIMD extensions on different general purpose processors.

Architecture	**Extension**	**SIMD Width** (Bits)
ARM	NEON	64-128
MIPS	MDMX	64
Scorpion	VeNum	128
Intel	SSE	128-256
AMD	3DNow!	128-256
PowerPC	AltiVec	128

SIMD extensions accelerate multimedia applications by exploiting data level parallelism (DLP). An important aspect with SIMD is that performance does not scale with increased media execution resources, as there are several bottlenecks in SIMD style media processing and that it is not possible to achieve significant additional performance improvement by making the processor wider to extract more parallelism. This is mainly due to overhead of supporting instructions that need to be significantly increased to scale with the increased SIMD width to keep the efficiency high.

For example, looking at table 1 a SIMD processor performing 16-bit operations on eight ways SIMD data will yield the best performance with 128 bits wide registers. At same time a processor designed for a coding standard that operates on 8x8 blocks with 8x8 transform matrices yield better performance with eight ways SIMD parallelism and 128 bits registers. This due to, eight ways SIMD parallelisms make full use of datapaths without cycles of unused datapath bandwidth. Also the registers within the processor must be large enough to hold a range of SIMD data values. An important issue to keep in mind is greater SIMD parallelism improves the average case but yields little performance improvements when processing a worst case stream. This depends on whether the applications used, for example if codec is based on 4x4 blocks then best performance is based on 64-bit registers. But in case of 8x8 blocks then 128-bit registers yield best performance. The ideal design keeps the processing elements fed

with data to avoid stalls, but if the design is wider than necessary this has no performance benefits. In the case of VLIW and SIMD both are high-performance and power-efficient designs but are usually well suited for only very specific types of application codes with large numbers of independent operations that can found by compilers or the programmer.

Another approach for getting better overall performance is through exploiting the instruction level parallelism (ILP) that exists in multimedia applications. There are many approaches, such as superscalarity or VLIW. Superscalarity means simultaneous execution of instructions on multiple parallel pipes in the processor. The main difference with other approaches, such as VILW, is allocating instructions to execution unit at run time. The added cost of superscalarity comes through utilization of hardware for achieving better parallel execution. There are two design approaches either an in-order or an out-of-order execution. The later is performance wise much better but is also more hardware complex. The out-of-order execution is crucial in getting the performance benefits of superscalar design as it makes runtime execution dependencies irrelevant, which was shown in paper IV [8]. An in-order superscalar processor on the other hand doesn't fully utilize the added hardware but still require more hardware complexity for a smaller performance benefits than is achieved by instruction level parallelism of a VLIW, which relies on the allocation of instructions to execution units at compile time. Superscalar out-of-order is also a direction which ARM processors have taken and in their Cortex-A9 architecture embraced this [11].

ILP can also be achieved, as stated above, through the use of VLIW. The mayor difference between the two designs is the hardware software trade-off. VLIW relies heavily on the compiler to effectively utilize the parallelism. This could be acceptable but it depends on the runtime execution predictability of the application. Multimedia applications and especially video codec's are highly runtime dependent making it difficult at compile time schedule optimal instruction execution order.

A direction that many chip vendors have taken in recent years is to offer parallel machines, or single chip multicore microprocessors. There are many reasons for this but one major reason being that in order for sustaining and keeping up with Moore's law scaling. There is up to certain level you can push a single core until the power consumption and core complexity gets too high. As the single core is pushed higher and higher in clock frequency the power required to do this grows in faster rate, leading to design that are very complex and power hungry. There are a number of different ways to define multicore and multi-processors. One definition is looking at the processing cores. The multicore system can be homogeneous multiprocessing as in symmetrical multiprocessors (SMP), where there are a number of identical cores tightly connected together. They can also be heterogeneous multiprocessors like in asymmetrical multiprocessor (AMP), where there are different type of cores connected via interconnect on the same die.

The main advantage of using multicore is that the required performance can come from increased number of cores compared to increase in frequency. However there are many challenges that need to be handled in order to get the promised performance. They range from designing multicore system, to efficiently writing parallel applications that utilize the system capabilities. Programming multicore system remains very challenging. The problem of how to take a piece of sequential code and optimally partition it across multiple cores remains unsolved [12]. A key factor is how well all the existing serial algorithms can be redesigned to take advantage of multicore systems. Algorithm designers need to understand the characteristics, including both advantages and limitations, of multicore to create algorithms that best match the platform.

Looking at video processing, an efficiently way for utilizing multicores is by using a single frame slicing that, for example, exists in H.264. By mapping the slices to a specific core a good distribution is achieved. If one on the other hand maps frames to different cores the communication and data synchronization overheads could lead to worst overall performance.

2.2.2 Audio processing

The other key applications in the multimedia domain are audio codecs and audio applications. Audio processing mostly focuses on different filter algorithms, such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Finite Impulse Response (FIR) and other digital signal processing filters. In the case of audio content, the MPEG-4 High-Efficiency Advanced Audio Coding v2 profile (HE-AAC v2) has proven to be one of the most efficient audio compression schemes [13]. This is one of the main codecs chosen for mobile handsets in 3GPP [14] and used in our research paper V [15]. The MPEG Advance Audio Codec (AAC) was designed to be the successor of the MP3 format. AAC generally achieves better sound quality than MP3 at similar bit rates. It is widely used and supported by almost all handheld devices.

How a decoder is designed and implemented is of significant importance especially in handheld devices. The computational complexity and memory requirements of the decoder are major factors in choosing the codecs in real-world applications. As these are directly related to power consumption and implementation cost. HE-AAC version 2 is a lossy data compression scheme for digital audio, consisting of three technologies. It is an extension of Low Complexity AAC (AAC LC) optimized for low-bitrate applications such as streaming audio. This is coupled with Spectral Band Replication (SBR), to enhance the compression efficiency in the frequency domain, and Parametric Stereo (PS) to enhance the compression efficiency of stereo signals as illustrated in figure 2.

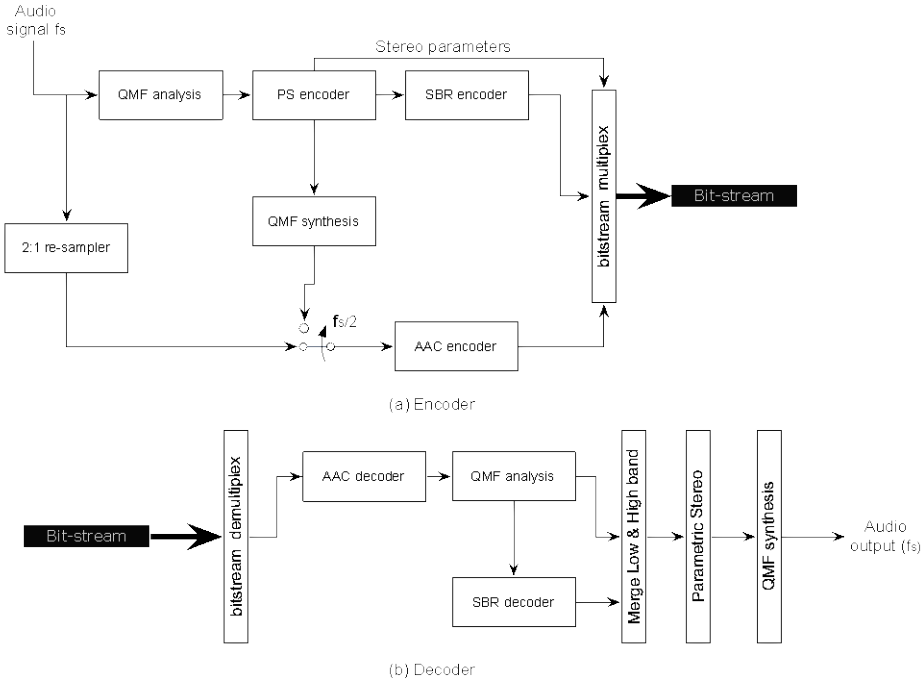


Figure 2. Block diagram of HE-AAC v2 encoder and decoder.

The main challenge in audio processing lies in finding the right balance between compression efficiency, audio quality and codec's computational complexity. By introducing SBR and PS the compression efficiency is significantly improved but this comes at expense of increased computational complexity for the HE-AAC decoder.

In most cases running audio applications could be performed in software on a general purpose processor, as even the most demanding audio applications with added post processing require less than 100 mega processor cycles per audio channel. Most often the addition of a VLIW or SIMD unit to the processor core provides even better computation power ratio, since audio applications have high degree of data parallelism. By increasing the number of audio channels the amount of processing workload increases quite linearly. For example, a true stereo audio has two times the workload of a single audio channel. This means audio workload can easily start adding up when the number of channels goes to 6 or even 8 surround. Another important difference between audio and video codecs, is that audio applications and codecs work in one dimension, making them easier to handle these applications in terms of processing and predictability.

2.2.3 Power constraints for processing

One of the technically fundamental constraints in embedded handheld systems is power or energy, since many of these systems are often battery powered [16]. This is closely related to the other constraint in embedded systems, performance. These two constrains, power and performance, often follow each other as in many cases they are closely correlated. They set the bases for design choices and trade-offs both in hardware and software. These design trade-offs range from choosing the silicon process, low power (LP) or high performance (G), to choosing different design libraries and hardware design tool settings. The design constraints bottom line is energy, how long operating time is directly connected to how much capacity there is in the batteries and how much power the system consumes for different use cases, which at the end determine end user experiences. The limitation in overall performance on the other hand is mainly due to silicon technology, die area and design choices (interconnect width, number of memory interfaces, etc.). On the power supply side, the slow increase in battery capacity in handheld portable devices has tightly constraint power consumption as shown in figure 3.

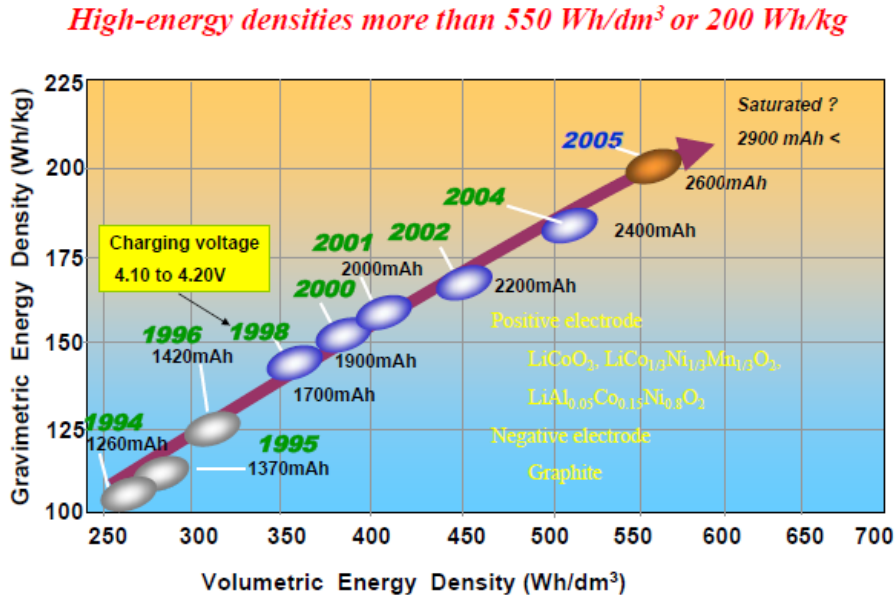


Figure 3. Energy density curve of lithium Ion battery development [17].

Looking at typical embedded handheld devices, such as Smartphones the typical operating voltage is around 3,7V. For different use cases the overall current consumptions vary significantly. For example, video playback uses 200-300mA, while

a voice call consumes 350mA. On top of this there are major power consuming peripherals, such as displays, cameras and different set of sensors. For example, a typical current consumption for a display is above 120 mA. Thus there is clear reason to distinguish between active power consumption (when the device is actively being used by an end user) and standby power consumption (when the device is in operational mode but the end user not actively working with the device). The standby typically have background activities running, such as keeping the cellular communication alive (~4mA), monitoring the subscription services, Facebook, email etc. (20-30mA) that the end user uses.

These parameters are essential in achieving the battery capacity constraints for handheld portable devices. Also, as the silicon have shrunk passed 65nm technology node, the overall leakage is quite significant. These lead to having a need to control and manage both dynamical as well as static power consumption. These include dividing the design in 10-20 different power domains in a hierarchical fashion, using clock and power gating, forward and reveres body biasing, and above all using dynamic voltage and frequency scaling. In this work, the main focus has been on dynamic power consumption for processing multimedia applications. Comparing the power benefits of SIMD enabled processors with non-SIMD processors [18], the main conclusion was that with added hardware for SIMD unit the power benefits through faster and more efficient execution, outweigh the added cost of the SIMD unit. This was true in terms of power consumption, hardware complexity and silicon area. Another major contributing factor for overall power consumptions is memory and external off-ship accesses which will be discussed in the next section.

Overall, looking at processing efficiency in terms of energy, depends thus on many factors, such as the chosen architecture and silicon technology, use cases the embedded device need to operate and how sophisticated power management is implemented in the system. For example, in many cases the peak power (up to 700-800 mW) is not the issue but rather for how long the sustained power consumption is the critical issue. This could be a major issue not only in terms of energy consumption for a battery operated device but also thermal limitations can become a limiting factor.

2.3 Multimedia Memory

One of the key factors that drives the cost and power dissipation of an embedded system on-chip is the memory architecture. The memory subsystem is a major contributor to the performance, power and area of a complex embedded system [19]. The memory subsystem can constitute a large part, up to 70% of the silicon area. This figure is expected to grow even further to over 90% by 2014 [20]. The main reason for this is the relative small design cost per area unit in terms of both manpower and time to market. Another reason is related to power consumption, since the heat dissipation per area unit is lower for memories than for logic. Thus, on-chip memory can be used to add functionality with smaller impact on system heat dissipation [21]. The key

challenge is to define a memory system that combines and satisfies the processor data requirements for executing efficiently, whilst minimizing cost and power consumption. In many multimedia intensive embedded systems, the overall area and power consumed by the memory subsystem (on-chip as well as off-chip) is up to 10 times greater than of processing, making memory a critical component of the design [22]. Thus, data placement and memory mapping is crucial for achieving good performance and optimal memory utilization. This can be achieved by optimally utilizing the on-chip memory, which can be SRAM, ROM and/or embedded DRAM/SRAM that is similar to off-chip DRAM.

In the field of memory architecture and memory management, there are two main approaches when dealing with multimedia applications. The first one uses dedicated zero wait state memories, often referred to as scratch pad memories (SPM). This approach requires often rewriting the applications to best fit these memory architectures. The second approach uses multilevel caches and this approach does not require rewriting the code but through careful optimizations based on a standard memory hierarchy, it is possible to get additional improvements.

The question that arises is how does the combination of the two memories, SPM and cache, behave and how should we best optimize and utilize the system, software and hardware. This is a typical trade-off, where we have a multidimensional optimization problem. In our work, presented in paper V [15], we introduce a methodology for mapping data and instructions to heterogeneous memory architectures consisting of SPM and multilevel caches. In that method, we define the problem as knapsack problem [23]. We show significant gains in lowering off-chip memory access by selectively mapping data and instruction to SPM, which provide an optimal solution for a given architecture. Our solution could either, minimize external memory accesses, minimize execution cycles or maximize SPM accesses.

Common for audio and video applications memory requirements are that they often work on streams of data. In the audio case the memory accesses are very regular, meaning one can predict the next data read quite accurately. In the video case this much more difficult, as the runtime dependency makes the data accesses much more irregular. Also, the amount of data being processed in video applications is at least an order of magnitudes larger compared to audio applications. A straightforward approach cannot be taken with video applications, as can be done with audio, especially due to memory bandwidth limitations in embedded systems. On top of this, in the video case, the encoding is execution dependent, meaning that selection of data for processing cannot be done at compile time. Instead the focus can be shifted to instructions of the algorithm and deciding what instruction sequences should be mapped to specific memory, such as SPM at compile time.

The SPM is organized into multiple memory banks, preferably in smaller memory banks, as these consume lesser power per access than larger memories. This presents an opportunity to place most frequently accessed data in smaller bank sizes. To obtain

good performance and few memory stalls, the data buffers of the application need to be placed carefully in different types of memory. This is typically done manually and hence takes a significant amount of time, up to several man months.

Memory management for multimedia applications especially video applications is thus an important area to focus on as huge amount of data is being processed. The result is heavy congestion of the memory subsystem, with the potential of breaking the system. Thus, there is a need to focus on algorithms for reducing amount of processed data as much possible, especially for encoding and decoding multimedia streams. This results in the situation where not all data is relevant for processing. Instead only a small portion of data is useful for processing and the rest can be disregarded. For example, the motion estimation algorithm is very selective in choosing the right data for processing, which leads to reduction in applications bandwidth needs.

Another key area when dealing with memory and multimedia applications is the applications themselves. Researchers have too often used non optimized reference code, where the overall behavior of the application is not the same as for the optimized code. For example, if a critical part of an application, which was computationally complex with high memory utilization and took up to 50% of the overall execution time, in the non optimized reference code, in an optimized real application, this part has been reduced to less than 15%. We showed this in paper III [18], where we compare real life code with standard codecs often used in research papers proposing solutions to speed up these applications.

2.3.1 Memory bandwidth requirements

As semiconductor process technology shrinks, it is feasible to design chips with greater data processing capability. The data rates and bandwidths for transferring data between chips do not increase at the same rate as data processing requirements. This result in performance of processors on video applications is constrained by memory bandwidth to off-chip DRAM memory devices in ever more applications. Bandwidth requirements for video applications are typically high and will further increase as resolutions, bit depths, and frame rates increase. For example, a SD video stream of 25fps consumes 15.6MB/s and for HD RGB12bit stream at frame rate of 120Hz consumes 1120 MB/s.

There are three different accesses types to off-chip memory: Bit stream (encoder writes, decoder reads), uncompressed frames (encoder reads, decoder writes), and reads of stored frame buffer(s) data for motion estimation or compensation. Because the bit stream is compressed, it consumes a negligible amount of bandwidth in most video applications. Reading and writing the uncompressed frame has significant impact on the memory bandwidth, resulting in a need for lowering it. This is one of the main reasons why the reads required for motion compensation or good motion quality motion estimation is essential as they account for most of the bandwidth consumed in typical video processor. As mentioned previously most dedicated hardware solutions

for video processing use full search block matching algorithm (FSBMA) leading to high memory bandwidth utilization.

The storage requirements and data access patterns of video functions have huge impact on system design and architecture. For example, most video codecs employ temporal algorithms, which require access to previous frames when processing a frame. This is important as it directly affects the memory storage needs. Required frame memory for HD H.264 decoding is around 15MB (YUV4:2:0, 8bit) which is not cost effective to have as on-chip memory of around $7mm^2$ per MB in 65nm technology. This gives a total area of $105mm^2$ for 15MB. Embedded DRAM would be alternative as it has higher density than SRAM but requires additional mask layers for manufacturing which adds to cost.

The bit stream accesses are both regular and irregular, depending what functional part of the codec is in use. For example, scaling and rotation have regular access patterns. Whereas other parts, such as decoding, encoding and interlacing use motion compensation, which is technique used for block prediction, has very irregular access pattern. This result in caches cannot alone reduce memory bandwidth to off-chip memory. For example, the motion compensation traffic in H.264 decoding is typically 100-200MB/s for SD resolution and 500-900MB/s for HD resolution content. For SoCs handling video content the typical accumulated memory bandwidth requirements is around 1-6GB/s. The processors are typically latency sensitive but by using a side buffer, such as SPM etc. can alleviate this latency sensitivity.

The importance of external memory bandwidth reduction cannot be over emphasized. For example, in video compression algorithms there are there type of frames intra coded frames (I), predicted frames (P) and bi-directional frames (B). For video encoders that perform motion estimation search for P and B frames, it greatly reduces off-chip memory bandwidth if an on-chip level-2 cache is implemented to store not only the current SAD block but also surrounding data. Motion search algorithms tend to have a large amount of overlap between successive or parallel SAD operations. So by having the adjacent data to SAD block it will often eliminate the need to access off-chip data for the next SAD. Coding video with I frames instead of P and B frames requires less compression processing and fewer off-chip memory accesses both of which save energy, but leads to significantly larger size bit streams.

Taking into consideration both memory constraints and media constraints, a need for a systematic methodology for memory management is required. In this thesis, a method for mapping of a heterogeneous, SPM and multilevel cache is proposed in paper V [15]. The applications used are both a video (H.264) and an audio (eAAC+) application.

Chapter 3

Related work

3.1 Processing

In the broad area of multimedia processing there has been made huge contributions. It has meant taking on the problem of processing from many different angles, from direct approaches, such as focusing on processing elements, to different hardware parallelizations. Other indirect approaches for enhancing multimedia processing work on limiting bandwidth requirements, optimizing memory architecture and interconnect/bus design, software enhancements, algorithmic improvements, compiler enhancements and overall system improvements.

Now focusing on work done on the processing unit for multimedia, these can range from fully programmable, to configurable and to hardwire solutions. In this work, we focus mainly on programmable cores but will give some remarks on other architectures as well. The programmable core can either be a general purpose CPU [24,25,26,27,28] or a digital signal processor DSP or very long instruction word VLIW, such as in [4,29,30,31,32,33,34,35,36,37,38] or a GPGPU general purpose graphical processing unit [39,40,41,42,43,44,45,46].

In many of studies using CPUs, these are combined with SIMD unit either as extensions to the main processor through ISA enhancements, much like NEON, MMX, Altivec [24,26,27], or use coprocessors [25,29]. The main difference is that in the later case the SIMD unit is not as tightly coupled with CPU as it is the case with ISA extensions. This is also an architecture often used when the focus is not on processing unit, but rather something else, such as special memory architectures [7] or software and complier enhancements [47]. In [26], the authors evaluated execution characteristic of multimedia applications on ARM architectures enhanced by SIMD. The authors of [24] focus on different media benchmark kernels and applications, such as DCT, motion estimation kernel, speech, and jpeg encoding applications. They use the results from their evaluations to propose architecture for the media applications they have

selected. They report good SIMD utilization for their applications but there is a need to take care of parallelism existing outside main loops and kernels. In their conclusions, they state that conventional ILP techniques need at least 8 or 16-way superscalar processor to provide improvements. The work in [25] focuses on proposing a low power mobile applications chip for Full-HD multi standard video codec. This design includes not only CPU but also DSP and streaming processor for execution of parts of the video codec. The target is to combine flexibility, provided by the CPU, and streaming processor for higher performance, in a heterogeneous multiprocessor architecture.

The other main approach is using DSP or VLIW processors for multimedia processing. These could also be combined with SIMD unit as in [4,29,31,33,37]. In [29] a 3-way 128bits VLIW is combined with a SIMD style VCP vector co-processing unit with three asymmetric parallel pipelines used for SAD, mean calculations and other kernel filter calculations. The VCP provides a performance boost that can handle up to HD video processing. The work presented in [30] proposes a new instruction set architectures based on variable VILW (32bit-128bit). The aim is to combine high performance with general purpose programmability. In [31] focus is on mapping motion estimation algorithm to a VLIW style DSP. The aim of [32] is to use a parallel Kahn process network KPN model on a VLIW DSP for H.264 encoding enhancement by low level algorithmic optimizations (motion compensation and estimation) with focus on potential instruction level parallelism ILP provided by the compiler. A vector micro-SIMD VLIW architecture is proposed in [33] for H.264 kernel, where the DLP regions of video encoder are improved so that the ILP regions become dominant. In [34], a reconfigurable function unit (RFU) is added to a VLIW architecture is proposed and evaluated for video compression speedups using a reference code. [4] presents a coprocessor chip for mobile baseband chips, using among others DSP to control a set of hardwired functional units for H.264 video compression. The combination of DSP and hardwired processing units gives good flexibility and is nice trade-off between the two extremes, with only using fully programmable verses hardwired design. In [36], a system on chip SoC is proposed and implemented. It combines two multimedia DSPs with CPU for video compression applications. A high performance VLIW DSP architecture is proposed, where the main feature is a flexible data path for using general purpose registers and SIMD units.

Another more resent approach has been to combine CPUs with GPUs for video processing and compression. In [44], they propose a approach where CPU and GPU work in parallel to accelerate video decoding in the PC environment. The GPU is used for part of the motion compensation while the rest of video decoding is done in the CPU. In [43] the focus is on the motion estimation search algorithm which one of the most important differentiating areas between a good and a great encoder. In [45,46] they propose using GPU for improving video encoding performance and at same time deriving the algorithms to be more suitable for GPU implementations. This is an ongoing trend, where the more recent work [48,49] aims to focus on GPU only rather

than investigating the optimal partition between GPU and CPU. But one important limitation with GPUs is that they can only perform identical computation concurrently on different stream processors. This makes overlapping different computations (prediction and reconstruction) very difficult [50]. In [51], they instead work on inter-frame parallelization which is much more suitable for GPUs. In [52] proposes new image analysis algorithms for parallel implementation on GPUs. Another approach for using GPUs and CUDA compute unified device architecture a multi-threaded programming model (hides the architecture from programmer) is used in [42,53,54] for video motion estimation and motion compensation is presented. This is a programming model proposed by nVidia [55] is gaining in popularity as it make the GPGPU much easier to program. Open Computing Language (OpenCL) [56] is the wide industry supported framework for programming for heterogeneous architectures consisting of GPU, CPU and other processor. There are a number of different mobile GPU architectures, such as ARMs Mali [57] and Imagination Technologies POWERVR SGX cores [58], supporting OpenCL,

Hardwired solutions were one of the most commonly used approaches for video encoding. It still gives the best performance per mm^2 silicon compared to any other solution. But due to lack of flexibility for updating to new standards and adding new feature enhancements it is no longer the preferred approach. But as stated previously this, when combined with CPU or other programmable cores for critical parts, is still competitive. In [59], the authors propose and implement a hardwired block for variable block size motion estimation. The chosen approach is to use full search block algorithm (FSBMA), which is a very common approach for hardwired solution. This basically means, the search algorithm goes through all possible macroblocks in a search window. This is however very inefficient as it means not only huge amount of processing, but also an even worse memory bandwidth load. This contributes to increased power consumption and decreased bandwidth efficiency. This is an area where authors of [10] try to address by proposing an approach that uses a two-dimensional systolic processor array to calculate absolute differences and a two-dimensional adder tree to sum up the absolute differences. The memory bandwidth problem is solved by scheduling data traffic. In [60], the proposed architecture is to combine the SIMD style hardwired units with CPU where a gradient descent search algorithm is implemented which reduces external memory accesses and lowers bandwidth requirements considerably.

Another set of approaches closely related that have been proposed for multimedia processing are extensible embedded processors and reconfigurable computing. In the case of extensible embedded processors there are commercial processors cores, such as Tensilica [61], ARC [62], Synopses former CoWare/LisaTek [63], which are available for usage in different designs. Application specific instruction processors (ASIP) are discussed in [64]. The main aim is to automatically detect and generate special instructions (SIs) for application speedup and/or power efficiency from the application code. An example for such work, is presented in [65] where a framework for dynamic

reconfiguration of application specific custom instructions is proposed. In [66,67] a method for generating SIs from profiling patterns is presented. The authors in [68], focuses on design space exploration with tool-supported connection to reconfigurable hardware for ASIPs. A constraint programming based method for modeling and solving scheduling and instruction selection for processors extended with a functionally reconfigurable cell fabric is presented in [69].

The reconfigurable architectures work either on full computational tasks using reconfigurable hardware, such as in [70] or on architectures combing reconfigurable parts with CPUs systems mainly focused on design-time predefined reconfiguration decisions [71,72]. But these are not suitable for dynamic runtime dependent systems, where the computational requirements/constraints change during runtime and are unpredictable during design time. An overview for reconfigurable architectures is summarized in [73]. In [74] a hardware design of an adaptive self-reconfigurable video processing platform is presented where the proposed architecture uses FPGAs to increase the design's flexibility. The authors in [75] presents an automated bus matrix synthesis flow for efficient transaction-level design space exploration of communication architecture in a reconfigurable multimedia system-on-chip platform. The main focus is on hardware interface selection where a method for using static analysis of communication behavior is proposed. [76] details a run-time task assignment heuristic algorithm that performs task assignment in a multiprocessor system-on-chip containing fine-grain reconfigurable hardware tiles. The basic idea is to map soft IP cores into reconfigurable hardware fabric which executes different task assignments. The algorithm is capable of managing a configuration hierarchy which improves the task assignment performance.

The work that has been done in multiprocessing and multicore with focus on multimedia can be divided into homogeneous architectures and heterogeneous architectures. In the field of embedded system, many of the system presented over the years have been heterogeneous multicore as they have had a CPU, DSP, different set of accelerators and GPU [77,78,79,80,81]. So it very much depends on how multicore is defined. For example, just taking the single GPU, this in itself is a multicore, consisting of as many as hundreds of smaller cores [55]. Taking a broader view of multicore, a good overview of the subject is given in [82]. A good survey of multicore is presented in [83], where different approach and issues are presented. The impact of multicore for multimedia applications can be focused on from different angles. In [27], the aim is to parallelize speech recognition on mobile multicore system. They also discuss the importance of on-chip level2 cache, when comparing and ARM architecture including on-chip level2 cache with Intel architecture with external level2 cache.

There is also significant number of studies done using homogeneous architectures, such as CELL [84] for multimedia [85,86,87,88,89]. Another equally important area is software and compiler for multicore. In [47], they present existing software compilation tools for both dynamic (just-in-time (JIT) compilers or dynamic optimizers)

and static compilation that locate and exploit different type of parallelism that can be utilized by multicore architectures. In [90], the authors describe different techniques for parallelizing video processing kernels for multicore systems. The article gives an overview of different parallel programming models, such as CUDA, OpenMP, OpenCL and Clik. In [86], the focus is on software design flow and proposes a move to higher levels of abstraction during software development. For multimedia applications, design flows based on the Kahn process network (KPN) model of computation should be used to fully exploit the multicore architectures.

3.2 Memory

There has been significant work done in the area of memory design and data management for embedded systems for multimedia applications. The main focus has been in trying to get most out of the system from the resource constraints architectures we have in embedded systems. The different approaches range from using caches, both single level and multilevel, to dedicated memories, such as SPMs.

Many papers have focused on improving data reuse in caches by using loop transformations [87,91]. In [9], a technique to organize data to best fit caches is presented in order to reduce external accesses. A closely related approach, for improving data locality for multimedia applications by introducing specific algorithms that data is placed in memory in such a way that the cache locking will be maximally respected is presented in [7]. Yet another set of approaches, for optimal cache usage is to modify the execution order of instructions to improve the cache performance of the multimedia applications, is presented in [6,92,93,94]. In [95] loop blocking is presented to improve cache performance but this has limited affect for multimedia applications due to significant amount of cache capacity misses still present.

Modern computer systems often provide memory system with non-uniform memory success times. Right placement of instructions and data for such systems can significantly improve program performance [96]. The data layout organization presented in [9], aims to reduce cache misses by arranging data in main memory. They place data at particular address block depending on lifetime and size. Thus, controlling the mapping of data to caches and hence removing influence of associativity on data mapping. The main limitation with these techniques is they do not take into account sharing caches with other applications.

Array padding is another data layout approach presented in [7,97,98] which aims to reduce cache misses by reducing cross conflict misses. The main limitation with array padding is that it does not eliminate majority of conflict misses. Modern memory architectures use often multilevel caches. This has become a standard in high performance embedded domain. In [99], usage of execution driven approach is presented for optimizing memory mapping to multilevel cache architecture. It uses hardware performance counters to decide how and when to apply loop transformations, and then select optimal transformation parameters using genetic algorithms.

The other dominant approach, for improving overall memory performance in embedded domain for multimedia applications, is using dedicated memories, such as SPM scratch pad memory. The main architectures range from SPM only to heterogeneous architectures where both SPM as well as caches are present.

For approaches with dedicated usage of SPM together with multimedia applications there has been significant work done. The focus has been to rewrite significant portions of applications to fit them to proposed architectures as shown in [9,100,101]. In [100], the focus is on an approach for optimization of usage of instruction SPM to reduce energy and increase performance. The aim of [101] is to introduce a systematic technique which assists in locality optimizations by selecting inlining functions that have strong data coupling between them, resulting in lowering external accesses. Other studies have looked at data mapping on SPM, both statically [102,103,104] and dynamically [105,106]. Common approach to these prior works has been mainly based on code transformations, i.e. loop and trace analysis and rewriting of applications for optimization.

There has also been work done on combination of compile time and runtime approaches, is proposed in [107]. This is done by inserting custom instructions to inform hardware to control data placement. Another approach focuses on managing memory space for different application in a SPM-only architecture and thus eliminating caches totally [104]. In this approach, the amount of data to be allocated to each application is decided based on the data reuse each application exhibits. Finally, in [108] a decoupled multilevel SPM-only architecture is presented. The architecture exploits mainly parallelism between address computation and processing the application data. Where an access processor calculates the addresses of data in memory hierarchy and the execution processor calculates the data.

The final approach is heterogeneous memory architecture where a combined architecture of SPM and cache is utilized, e.g. mapping of data and instruction to SPM. Here the main question is how to optimally utilize this architecture. The main drawback with static approaches is the limitations to exploit the dynamically changing data access patterns of programs. Most of the previous work is based on static allocations [6,7,9,87,91,92,93,94,95,97,98]. The dynamic memory mapping approaches are mostly profile based. The main disadvantage of these techniques is the difficulty in acquiring reasonable profiles for other than small applications or kernels. Another dynamic approach is the introduction of techniques that requires hardware support [100,104,105,106,107,108], with the main drawback of making them less applicable due to added silicon costs and increased in flexibility and complexity. In [19] the layout problem is solved for an on-chip memory architecture that has both SPM and cache memory. Looking at other non CPU architectures, in [109] they address partitioning of simultaneously accessed data variables in DSP architectures using multiple single port memory banks to avoid memory conflicts.

The different memory mapping optimization techniques are based on different algorithms, such as genetic algorithms, heuristic algorithms and constraint programming. In [99], a framework for memory mapping is presented where the selection of transformation parameters uses genetic algorithms. In [22], genetic algorithm is used for mapping objects to SPM only memory architecture for DSPs. The method presented for data layout problem is view as a multi-objective genetic algorithm with performance and power being the cost functions. In [110] and [111] a memory hierarchy exploration is done using genetic algorithm framework. Their target architecture is cache only design with focus on a single formula which combines area, average accesses and power. The problem is modeled as a multi objective genetic algorithms problem. In [112] a multi objective frame work for memory architecture exploration is presented based on a combined genetic algorithm for the memory architecture exploration and a simple heuristic algorithm for data mapping to the memory architecture. This method focuses, as in [22], on DSP architectures using SPM only memory architecture as target for specific multimedia applications.

Chapter 4

Papers Survey

In first phase, paper I and II, we analyzed video encoder MPEG-4 together with initial study on processing data using SIMD. In second phase, papers III and IV, we carried out more in-depth analysis and more detailed study on video encoders MPEG-4 and H.264. From our study we saw two areas standout, heterogeneous memory architectures to satisfy multimedia applications different memory needs and instruction level parallelism (ILP) for handling the increased amount of control instructions in multimedia applications. In third phase, paper V we explored the memory architectures and focus on trade-off that are need to be made when handling video H.264 and audio eAAC+. In this work we proposed a methodology for mapping the optimal memory configuration for a certain multimedia application. The method is based on a combination of simulation as well as analysis using constraint programming (CP), for finding non-dominated solutions (Pareto points) for a given memory configuration.

Evaluation of SIMD Architecture Enhancement in Embedded Processors for MPEG-4

Paper I studies the effects of using SIMD processor extension, developed to enhance the processor performance, for streaming applications. An extensive evaluation of the proposed architecture extension, using Simplescalar simulator, was performed, showing that it is possible to achieve high performance with acceptable power consumption.

Analysis of Embedded Processors for Streaming Media Applications

Paper II analyzes multimedia performance of an embedded processor family used in most of wireless handheld devices, using full video MPEG-4 encoding application with optimized algorithms for mobile devices. There are significant performance improvements when using new architectural solutions, such as Single Instruction Multiple Data (SIMD) extensions and increasing clock frequency, enabled by deeper instruction pipeline, in embedded processors. In order to provide enough performance for MPEG-4 encoding there is a need to have a more aggressive SIMD architecture with wider data paths. This introduces some new issues among others a memory bandwidth bottleneck.

Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors

Paper III investigates the impact of different memory configurations (memories and multilevel cache architectures) together with SIMD extended embedded processor on performance and energy consumption of the video encoding applications, MPEG-4 and H.264. The results show that the use of the standard cache-based architecture achieves almost the same performance as SIMD dedicated memory architecture for full video encoding applications, against common belief. This makes it difficult to justify using dedicated memory for this kind of embedded systems, when energy consumption and cost of implementation are also considered.

Performance Improvement for H.264 Video Encoding using ILP Embedded Processor

Paper IV examines the impact of instruction level parallelism (ILP) and tradeoffs in superscalar performance for full H.264 video encoding application and gives quantitative performance measures of a superscalar architecture. Most research efforts have concentrated on data intensive parts, such as kernels but these are taking less time from the entire execution as encoders are using new, more efficient algorithms. This important fact cannot be neglected since new video encoding standards have been proposed and the amount of other than data intensive computations has increased significantly. There is significant improvement for the entire application when using superscalar architecture with out-of-order execution scheme.

Design Space Exploration for Optimal Memory Mapping of Data and Instructions in Multimedia Applications to Scratch-Pad Memories

In paper V, we propose a new methodology for optimal memory mapping of data and instructions to Scratch-Pad Memories (SPM). The optimization is done by finding Pareto points, using multi-objective optimization that combines different cost functions. Our proposed methodology provides a way to combine SPMs with caches to optimally use this memory architecture. By using this method there is only a need to perform limited number of simulations, instead of performing extensive simulation of all possible combinations. The methodology is intended to be used in real-life situations in industry where there is often a need for mapping third party applications to a specific architecture.

Chapter 5

Contributions

This thesis presents work on design space exploration for embedded systems with focus on multimedia applications, such as video and audio codecs. It addresses and presents solutions to problems related to system architecture trade-off, pertinent to memory and processing. Each of the following four points correspond to and presents the contributions made by the author of this thesis, through the constituent five papers, where each point is part of at least one of the five papers included in the thesis. The author of this thesis is the primary author of all the included papers.

- *A new methodology for optimal memory mapping* of data and instructions to Scratch-Pad Memories (SPM). The optimization is done by finding Pareto points, using multi-objective optimization that combines different cost functions. The proposed methodology provides a way to combine SPMs with caches to optimally use this memory architecture. By using this method there is only a need to perform limited number of simulations, instead of performing extensive simulation of all possible combinations. The methodology is intended to be used in real-life situations in industry where there is often a need for mapping third party applications to a specific architecture.
- *Quantitative performance assessment for superscalar architecture with instruction level parallelism (ILP) for video encoding applications.* Different trade-offs for full H.264 video encoding has been evaluated. Most research efforts so far have concentrated on the data intensive parts, such as kernels but these are taking less time from the entire execution as encoders are using new, more efficient algorithms. This important fact cannot be neglected since new video encoding standards have been proposed and the amount of other than data intensive computations has increased significantly. There is significant

improvement for the entire application when using superscalar architecture with out-of-order execution scheme.

- *A vectorized architecture with ISA extensions for multimedia applications is proposed and evaluated* by implementing SIMD architecture solution on an embedded processor. The proposed architecture is tightly coupled with the micro-architecture of the processor and design to handle up to 128-bit wide arithmetic operations.
- *Performance and energy consumption evaluation for different memory configurations for SIMD extended embedded processor.* We consider dedicated memories and multilevel cache architectures. The results show, against common belief, that the use of the standard cache based architecture achieves almost the same performance as SIMD dedicated memory architecture for full video encoding applications. This makes it difficult to justify using dedicated memory for this kind of embedded systems, when energy consumption and cost of implementation are also considered.

Chapter 6

Conclusions and Future Trends

An embedded system for multimedia is a reality today. One of the main challenges for optimized use of the embedded systems for multimedia lies in memory architecture. In this paper, we have focused on SIMD extended single general-purpose processor design with heterogeneous memory architectures. The main conclusions from this work are that on a system level heterogeneous memory architecture works very well for multimedia applications. This is mainly because the heterogeneous architecture very effectively can combine different set of requirements concurrently. Optimal utilization of this architecture is the challenge.

The constraint-based method proposed in this thesis finds an optimal data and instruction mapping for multimedia applications. This method systematically analyzes and finds optimal memory mapping for a given application and architecture. Using this method there will be need to only perform limited number of simulations, instead of performing extensive simulation of all possible combinations. Our results indicate high accuracy for predicting memory accesses, above 90%. This methodology can be adapted quite easily to real industrial situations where there is often a need for mapping third party application to a specific architecture. This method can be extended to other cases of multidimensional optimizations.

Data level parallelism (DLP) but also instruction level parallelism (ILP) is important to consider during processing of multimedia applications. SIMD extended general-purpose superscalar processor architecture provide both DLP as well as ILP. There are of course other approaches that can be utilized, such as multicore and VLIW architectures.

Future embedded systems are evolving based on different trends, CPU architectural evolution and Memory optimization architecture. CPU architecture evolution is where ore complex core (deep pipelining, advance branch prediction and superscalar out-of-order) as well as increase number of tightly coupled cores in multicore configuration.

The memory architecture on the other hand is to address challenges connected to memory architectures, such as reducing bandwidth bottlenecks, hiding latency, optimal usage of heterogeneous memory architecture etc. Most of the issues are possible to be solved using the architectural enhancements based on future memory integration technologies, such as 3D die stacking with huge bandwidth potential. It can open new opportunities for new system design and exploration. The techniques used are either micro-bumps or through-silicon-vias (TSV) also known as wide-IO. These memory integrations enable large number of connections, with possibility of many interfaces. The introduction of TSV brings a totally new dimension to embedded system architecture, where memory bandwidth is no longer the bottleneck it used to be. There are still many technical and non-technical challenges with these technologies, such as manufacturing, standardization and costs, before being widely used in embedded systems.

Multi-core embedded systems are today leading processing technology, how best to utilize the processing abilities and how to leverage these high performing cores. Multi-cores will not provide single thread peak performance but, the advantage of multicores lies in features such as vector units and out-of-order superscalar architectures with advance speculative execution.

To make these multicores the more acceptable processors in embedded systems the biggest challenge is keeping the power consumption at the minimum with exponential increase in performance.

The trend for muticore system is nothing new in embedded systems as this has been the path early on. The CPU+DSP approach was the chosen architecture as early as end off 1990s. The new is symmetrical multi processing (SMP), a homogeneous tightly connected multicore system, in contrast to the older loosely coupled multicore designs. Combined with the rest of the system, graphical processing unit (GPU), digital signal processor (DSP), hardwired accelerators and other peripherals on the same die, build a well-tuned and balanced system on chip. Connecting it all together, a high-speed interconnection network on chip is often used. The challenge lies in keeping to the power and size constrains at same time harvesting the desired performance. One such limitation is Amdahl's law, which states that the parallel speedup is limited by the serial code in a program.

Advance embedded system on chip design, with TSV enables the possibility of removing the memory bottlenecks and opens up for more performance. Short term, the removal of memory as a bottleneck will push the performance higher with the overall performance of the processing elements (CPU, GPU and IP blocks) increasing proportionally. Long term, these improvements could have significant influence of system architecture as well processor design (SoC). In a 3D silicon structure, memory and logic units can be designed to work more integrated such that performance increases can be achieved. At same time many new challenges arises, such as for example, optimally finding the best placement of processing and memory units or even

mapping of different applications to these 3D architectures. Thus a systematic design space exploration method similar to the constraint based method proposed in this thesis can be applied for optimally exploiting these 3D architectures.

Multimedia is the most prominent use case for all handheld devices, with LTE bandwidth increasing exponentially, lower costs per bit, and higher quality of service [113]. All these advancements together with more complex signaling algorithms [114] enables increase in multimedia workload significantly 10x [2]. This comes at the same time as increase in display resolution up to and beyond HD [25] leading to even higher workload for multimedia. Another direction that multimedia workloads (audio as well as video) are taking is focusing on quality instead of compression. This is a combination of both the increased bandwidth with newer cellular standards as well as more processing performance of the handheld devices.

Bibliography

[1] K. Kuchcinski, "Constraints-driven scheduling and resource assignment," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 8, pp. 355--383, 2003.

[2] C. H. van Berkel, "Multi-core for mobile phones," *2009 Design, Automation&Test in Europe Conference&Exhibition*, pp. 1260-1265, 2009.

[3] ARM Ltd. (2010, Aug.) NEON.
[Online]. http://www.arm.com/products/processors/technologies/neon.php

[4] J. Meehan, "Media processor architecture for video and imaging on camera phones," *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5340-5343, 2008.

[5] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore processors," *IEEE Signal Processing Magazine*, vol. 26, pp. 26-37, 2009.

[6] E. De Greef, *Storage Size Reduction for Multimedia Application.* Leuven: IMEC, 1998, Phd thesis.

[7] P. R. Panda, N. D. Dutt, and A. Nicolau, "Memory data organization for improved cache performance in embedded processor applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 2, pp. 384--409, 1997.

[8] A. R. Iranpour and K. Kuchcinski, "Performance Improvement for H.264 Video Encoding using ILP Embedded Processor," , 2006, pp. 515--521.

[9] C. Kulkarni, C. Ghez, M. Miranda, F. Catthoor, and H. de Man, "Cache conscious data layout organization for embedded multimedia applications," , 2001, pp. 686--693.

[10] D.-X. Li, W. Zheng, and M. Zhang, "Architecture Design for H.264/AVC Integer Motion Estimation with Minimum Memory Bandwidth," *IEEE Transactions on Consumer Electronics*, vol. 53, pp. 1053-1060, 2007.

[11] ARM Ltd. (2007, Sept.) The ARM Cortex-A9 Processors ARM Ltd. White Paper. [Online]. http://www.arm.com/pdfs/ARMCortexA-9Processors.pdf

[12] L. J. Karam et al., "Trends in multicore DSP platforms," *IEEE Signal Processing Magazine*, vol. 26, pp. 38-49, 2009.

[13] ISO/IEC, "HE AAC version 2," ISO/IEC , Standard 14496-3:2005/Amd.2, 2005.

[14] The 3rd Generation Partnership Project (3GPP) unites telecommunications standards bodies. (2010, July) 3GPP specification releases. [Online]. http://3gpp.org/Releases

[15] A. R. Iranpour and K. Kuchcinski, "Design space exploration for optimal memory mapping of data and instructions in multimedia applications to Scratch-Pad Memories," *2009 IEEE/ACM/IFIP 7th Workshop on Embedded Systems for Real-Time Multimedia*, pp. 89-95, 2009.

[16] T. Mudge, "Power: A First-Class Architectural Design Constraint," *Computer*, vol. 34, pp. 52--58, 2001.

[17] T. Ohzuku, "An Application of Lithium Insertion Materials to 12 V lead-free Batteries," Electrochemistry and Inorganic Chemistry Laboratory,Graduate School of Engineering,Osaka City University (OCU), Osaka, Presentation 2007.

[18] A. R. Iranpour and K. Kuchcinski, "Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors," , 0 2006, pp. 309--320.

[19] P. R. Panda, N. D. Dutt, and A. Nicolau, "On-chip vs. off-chip memory: the data partitioning problem in embedded processor-based systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, pp. 682--704, 2000.

[20] ITRC. (2010) International Technology Roadmap for Semiconductors. [Online]. http://public.itrs.net

[21] E. Schmidt, *Power Model of Embedded Systems, PhD Dissertation*., 2003.

[22] T. S. Kumar, C. P. Ravikumar, and R. Govindarajan, "MODLEX: A Multi Objective Data Layout EXploration Framework for Embedded Systems-on-Chip," , 2007, pp. 492--497.

[23] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*.: John Wiley & Sons, Inc., 1990.

[24] D. Talla, L. K. John, and D. Burger, "Bottlenecks in Multimedia Processing with SIMD Style Extensions and Architectural Enhancements," *IEEE Trans. Comput.*, vol. 52, pp. 1015--1031, 2003.

[25] K. Iwata et al., "A 342 mW Mobile Application Processor With Full-HD Multi-Standard Video Codec and Tile-Based Address-Translation Circuits," *IEEE Journal of Solid-State Circuits*, vol. 45, pp. 59-68, 2010.

[26] V. Parthasarathy, S. A. Bharathi, and V. R. Uthariaraj, "Performance Analysis of Embedded Media Applications in Newer ARM Architectures," *Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on*, pp. 210-214, 2005.

[27] S.-M. Cho, D.-W. Im, and H.-J. Song, "Parallelization and Analysis of Speech Recognition on Mobile Multi-Core Processor," *2009 6th IEEE Consumer Communications and Networking Conference*, pp. 1-2, 2009.

[28] K. Diefendorff and P. K. Dubey, "How multimedia workloads will change processor design," *Computer*, vol. 30, pp. 43-45, 1997.

[29] T. Wada et al., "A VLIW vector media coprocessor with cascaded SIMD ALUs," *IEEE*

Trans. Very Large Scale Integr. Syst., vol. 17, pp. 1285--1296, 2009.

[30] R. B. Lee and A. M. Fiskiran, "PLX: An Instruction Set Architecture and Testbed for Multimedia Information Processing," *The Journal of VLSI Signal Processing*, vol. 40, pp. 85-108, 2005.

[31] W. Lee, H. Choi, and W. Sung, "Algorithm and Software Optimization of Variable Block Size Motion Estimation for H.264/AVC on a VLIW--SIMD DSP," *Journal of Signal Processing Systems*, vol. 51, pp. 289-302, 2008.

[32] A. C. Ammari, A. Jemai, H. K. Zrida, and M. Abid, "ILP platform optimization of a YAPI parallel H.264/AVC encoder," *2008 2nd International Conference on Signals, Circuits and Systems*, pp. 1-6, 2008.

[33] E. Salami, "A Vector-µSIMD-VLIW Architecture for Multimedia Applications," , 2005, pp. 69--77.

[34] D. Rizzo and O. Colavin, "A video compression case study on a reconfigurable VLIW architecture," *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, pp. 540-546, 2002.

[35] I. Barbieri, M. Bariani, A. Scotto, and M. Raggio, "Multimedia terminal system-on-chip design and simulation," *EURASIP J. Appl. Signal Process.*, vol. 2005, pp. 2694--2700, 2005.

[36] H. Stolberg et al., "An SoC with two multimedia DSPs and a RISC core for video compression applications," *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, pp. 330-531 Vol.1, 2004.

[37] L.-D. Van et al., "A high-performance area-aware DSP processor architecture for video codecs," *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, vol. 3, pp. 1499-1502Vol.3, 2004.

[38] Y. Kun, Z. Chun, M. Songping, and W. Zhihua, "DSP architecture for motion estimation acceleration," *Solid-State and Integrated Circuits Technology, 2004. Proceedings. 7th International Conference on*, vol. 3, pp. 1609-1612 vol.3, 2005.

[39] Y.-L. Huang, Y.-C. Shen, and J.-L. Wu, "Scalable computation for spatially scalable video coding using NVIDIA CUDA and multi-core CPU," in *Proceedings of the seventeen ACM international conference on Multimedia, MM '09*, 2009, pp. 361--370.

[40] C.-H. Sun, K.-H. Lok, Y.-M. Tsao, C.-M. Chang, and S.-Y. Chien, "CFU: multi-purpose configurable filtering unit for mobile multimedia applications on graphics hardware," in *Proceedings of the Conference on High Performance Graphics 2009, HPG'09*, 2009, pp. 29--36.

[41] M. C. Kung, O. C. Au, P. H. Wong, and C. H. Liu, "Block based parallel motion estimation using programmable graphics hardware," *2008 International Conference on Audio, Language and Image Processing*, pp. 599-603, 2008.

[42] S. Momcilovic and L. Sousa, "Development and evaluation of scalable video motion estimators on GPU," *2009 IEEE Workshop on Signal Processing Systems*, pp. 291-296, 2009.

[43] M. Schwalb, R. Ewerth, and B. Freisleben, "Fast Motion Estimation on Graphics Hardware for H.264 Video Encoding," *IEEE Transactions on Multimedia*, vol. 11, pp. 1-10, 2009.

[44] G. Shen, G.-P. Gao, S. Li, H.-Y. Shum, and Y.-Q. Zhang, "Accelerate Video Decoding With Generic GPU," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, pp. 685-693, 2005.

[45] Y.-C. Lin et al., "Multi-Pass Algorithm of Motion Estimation in Video Encoding for Generic GPU," *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4451-4454, 2006.

[46] C.-W. Ho, O.-C. Au, S. Gary Chan, S.-K. Yip, and H.-M. Wong, "Motion Estimation for H.264/AVC using Programmable Graphics Hardware," *2006 IEEE International Conference on Multimedia and Expo*, pp. 2049-2052, 2006.

[47] M. Mehrara et al., "Multicore compilation strategies and challenges," *IEEE Signal Processing Magazine*, vol. 26, pp. 55-63, 2009.

[48] G. Jin and H.-J. Lee, "A Parallel and Pipelined Execution of H.264/AVC Intra Prediction," , 2006, p. 246.

[49] W Lee, S Lee, and J Kim, "Pipelined Intra Prediction Using Shuffled Encoding Order for H.264/AVC," *TENCON 2006 - 2006 IEEE Region 10 Conference*, pp. 1-4, 2006.

[50] N.-M. Cheung, O.-C. Au, M.-C. Kung, P.-H. Wong, and C.-H. Liu, "Highly Parallel Rate-Distortion Optimized Intra-Mode Decision on Multicore Graphics Processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, pp. 1692-1703, 2009.

[51] K.-W. Yoo and H.-H. Kim, "Intra prediction method and apparatus," H04N7/34B EP1526738, Apr. 27, 2005.

[52] Y.-K. Chen, W. Li, J. Li, and T. Wang, "Novel parallel Hough Transform on multi-core processors," *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1457-1460, 2008.

[53] R. Rodriguez, J. L. Martinez, G. Fernandez-Escribano, J. M. Claver, and J. L. Sanchez, "Accelerating H.264 inter prediction in a GPU by using CUDA," *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*, pp. 463-464, 2010.

[54] W.-N. Chen and H.-M. Hang, "H.264/AVC motion estimation implmentation on Compute Unified Device Architecture (CUDA)," *2008 IEEE International Conference on Multimedia and Expo*, pp. 697-700, 2008.

[55] Nvidia Corp. (2009, February) NVIDIA CUDA Compute Unified Device Architecture-

Programming Guide.

[56] The Khronos Groupe. (2011, April) Khronos Groupe. [Online]. http://www.khronos.org/opencl/

[57] ARM Ltd. (2011, June) Mali-T604. [Online]. http://www.arm.com/products/multimedia/mali-graphics-hardware/mali-t604.php

[58] Imagnination Technologies Ltd. (2011, June) PowerVR Graphics. [Online]. http://www.imgtec.com/powervr/powervr-graphics.asp

[59] M. N. Bojnordi, M. Semsarzadeh, M. R. Hashemi, and O. Fatemi, "Efficient Hardware Implementation for H.264/AVC Motion Estimation," *APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1749-1752, 2006.

[60] Y. Kuroda et al., "A sub-mW MPEG-4 motion estimation processor core for mobile video application," *Design Automation Conference, 2004. Proceedings of the ASP-DAC 2004. Asia and South Pacific*, pp. 527-528, 2004.

[61] Tensilica Inc. (2010) Tensilica. [Online]. http://www.tensilica.com/

[62] ARC Inc. (2010) ARC® Configurable CPU/DSP Cores. [Online]. http://www.arc.com/configurablecores/

[63] CoWare/LisaTek. (2010) Synopsys. [Online]. http://www.synopsys.com/Tools/SLD/Pages/default.aspx

[64] J. Henkel, "Closing the SOC Design Gap.," *Computer*, vol. 36, pp. 119-122, 2003.

[65] H. Huynh, J. Sim, and T. Mitra, "An efficient framework for dynamic reconfiguration ofÂ instruction-set customization," *Design Automation for Embedded Systems*, vol. 13, no. 1, pp. 91-113, 2009.

[66] K. Atasu, L. Pozzi, and P. Ienne, "Automatic Application-Specific Instruction-Set Extensions under Microarchitectural Constraints," in *DAC '03 Proceedings of the 40th annual Design Automation Conference*, Anaheim, CA, USA, 2003, pp. 256--261.

[67] F. Sun, S. Ravi, A. Raghunathan, and N. K. Jha, "A scalable application-specific processor synthesis methodology," *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pp. 283-290, 2003.

[68] A. Chattopadhyay et al., "Design Space Exploration of Partially Re-configurable Embedded Processors," *2007 Design, Automation&Test in Europe Conference&Exhibition*, pp. 1-6, 2007.

[69] C. Wolinski, K. Kuchcinski, and E. Raffin, "Combined scheduling and instruction selection for processors with reconfigurable cell fabric," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 15, no. 1, December 2009.

[70] M. Ullmann, M. Huebner, B. Grimm, and J. Becker, On-Demand FPGA Run-Time System

for Dynamical Reconfiguration with Adaptive Priorities, 0 0, 2004.

[71] R. Gao, D. Xu, and J. P. Bentley, "Reconfigurable Hardware Implementation of an Improved Parallel Architecture for MPEG-4 Motion Estimation in Mobile Applications.," *IEEE Transactions on Consumer Electronics*, vol. 49, pp. 1383-1391, 2003.

[72] C. Wei and M. Z. Gang, "A novel SAD computing hardware architecture for variable-size block motion estimation and its implementation with FPGA," *ASIC, 2003. Proceedings. 5th International Conference on*, vol. 2, pp. 950-953, 2003.

[73] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Computing Surveys*, vol. 34, pp. 171-210, 2002.

[74] K. F. Ackermann, B. Hoffmann, L. S. Indrusiak, and M. Glesner, "Enabling self-reconfiguration on a video processing platform," *2008 International Symposium on Industrial Embedded Systems*, pp. 19-26, 2008.

[75] G. Lee et al., "Communication architecture design for reconfigurable multimedia SoC platform," *Design Automation for Embedded Systems*, vol. 14, pp. 1-20, 2010.

[76] V. Nollet, P. Avasare, H. Eeckhaut, D. Verkest, and H. Corporaal, "Run-Time Management of a MPSoC Containing FPGA Fabric Tiles," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, pp. 24-33, 2008.

[77] Sony Computer Entertainment America LLC. (2011, Feb.) Sony NGP. [Online]. http://us.playstation.com/ngp/

[78] Texas Instruments Inc. (2009) OMAP 4: Mobile applications platform. [Online]. http://focus.ti.com/lit/ml/swpt034/swpt034.pdf

[79] ST-Ericsson. (2011, Jan.) NovaThor™ U9500. [Online]. http://www.stericsson.com/platforms/u9500-nova-thor.jsp

[80] Qualcomm Inc. (2011, Feb.) The Inside Story. [Online]. http://www.qualcomm.com/snapdragon/specs

[81] Nvidia Corp. (2011, Jan.) What is GPU Computing? [Online]. http://www.nvidia.com/object/GPU_Computing.html

[82] W. Wolf, A. Jerraya, and G. Martin, "Multiprocessor System-on-Chip (MPSoC) Technology," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1701-1713, 2008.

[83] K. F. Faxén et al., "Multicore computing—the state of the art," , 2008.

[84] IBM Inc. (2010, May) Introduction to the Cell Broadband Engine. [Online]. https://www-01.ibm.com/chips/techlib/techlib.nsf/techdocs/D21E662845B95D4F872570AB0055404D

[85] N. Parakh, A. Mittal, and R. Niyogi, "Optimization of MPEG 2 Encoder on Cell B. E. Processor," *2009 IEEE International Advance Computing Conference*, pp. 423-427, 2009.

[86] W. Haid, K. Huang, I. Bacivarov, and L. Thiele, "Multiprocessor SoC software design flows," *IEEE Signal Processing Magazine*, vol. 26, pp. 64-71, 2009.

[87] D. F. Bacon, S. L. Graham, and O. J. Sharp, "Compiler transformations for high-performance computing," *ACM Comput. Surv.*, vol. 26, pp. 345--420, 1994.

[88] X. He, X. Fang, C. Wang, and S. Goto, "Parallel HD encoding on CELL," *2009 IEEE International Symposium on Circuits and Systems*, pp. 1065-1068, 2009.

[89] L.-K. Liu et al., "Digital Media Indexing on the Cell Processor," *Multimedia and Expo, 2007 IEEE International Conference on*, pp. 1866-1869, 2007.

[90] D. Lin et al., "The parallelization of video processing," *IEEE Signal Processing Magazine*, vol. 26, pp. 103-112, 2009.

[91] K. S. McKinley, S. Carr, and C.-W. Tseng, "Improving data locality with loop transformations," *ACM Trans. Program. Lang. Syst.*, vol. 18, pp. 424--453, 1996.

[92] M. Kandemir, J. Ramanujam, and A. Choudhary, "Improving cache locality by a combination of loop and data transformations," *Computers, IEEE Transactions on*, vol. 48, pp. 159-167, 1999.

[93] C Kulkarni, F Catthoor, and H De Man, "Hardware cache optimization for parallel multimedia applications," vol. 1470, pp. 923-932, 1998.

[94] D. Kulkarni and M. Stumm, "Linear Loop Transformations in Optimizing Compilers for Parallel Machines," *The Australian computer journal*, pp. 41-50, may 1995.

[95] M. D. Lam, E. E. Rothberg, and M. E. Wolf, "The cache performance and optimizations of blocked algorithms," in *Proceedings of the fourth international conference on Architectural support for programming languages and operating systems, ASPLOS-IV*, 1991, pp. 63--74.

[96] M. Brorsson, "Performance Impact of Code and Data Placement on the IBM RP3," IBM, Technical Report 1989.

[97] N. Manjikian and T. S. Abdelrahman, "Array Data Layout for the Reduction of Cache Conflicts," in *In Proceedings of 8th Int'l. Conf. on Parallel and Distributed Computing Systems*, 1995.

[98] P. R. Panda, H. Nakamura, N. D. Dutt, and A. Nicolau, "Augmenting loop tiling with data alignment for improved cache performance," *Computers, IEEE Transactions on*, vol. 48, pp. 142-149, 1999.

[99] P. Lu, Y. Che, and Z. Wang, "A Framework for Effective Memory Optimization of High Performance Computing Applications," *2009 11th IEEE International Conference on High Performance Computing and Communications*, pp. 95-102, 2009.

[100] A. Janapsatya, A. Ignjatovic, and S. Parameswaran, "A novel instruction scratchpad memory optimization method based on concomitance metric," in *In Proceedings Asia and*

South Pacific Conference on Design Automation, 2006, pp. 612--617.

[101] J. Absar, P. Marchal, and F. Catthoor, "Data-access optimization of embedded systems through selective inlining transformation," *Embedded Systems for Real-Time Multimedia, 2005. 3rd Workshop on*, pp. 75-80, 2005.

[102] J. Absar and F. Catthoor, "Analysis of scratch-pad and data-cache performance using statistical methods," *Design Automation, 2006. Asia and South Pacific Conference on*, pp. 820-825, 2006.

[103] A. Dominguez, "Heap data allocation to scratch-pad memory in embedded systems," phdthesis 2007.

[104] O. Ozturk, M. Kandemir, and I. Kolcu, "Shared Scratch-Pad Memory Space Management," in *Proceedings on 7th International Symposium on Quality Electronic Design (ISQED'06)*, 2006, pp. 576--584.

[105] S. Mamagkakis et al., "Custom design of multi-level dynamic memory management subsystem for embedded systems," *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pp. 170-175, 2004.

[106] D. Atienza et al., "Efficient system-level prototyping of power-aware dynamic memory managers for embedded systems," *Integr. VLSI Journal*, vol. 39, pp. 113--130, 2006.

[107] A Janapsatya, S Parameswaran, and A Ignjatovic, "Hardware/software managed scratchpad memory for embedded system," in *IEEE/ACM International Conference on Computer Aided Design, ICCAD-2004.*, 2004, pp. 370--377.

[108] A. Milidonis et al., "A Decoupled Architecture of Processors with Scratch-Pad Memory Hierarchy," *2007 Design, Automation&Test in Europe Conference&Exhibition*, pp. 1-6, 2007.

[109] R. Leupers and D. Kotte, "Variable partitioning for dual memory bank DSPs," in *Proceedings 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '01).*, 2001, pp. 1121--1124.

[110] M. Palesi and T. Givargis, "Multi-objective design space exploration using genetic algorithms," in *Proceedings of the Tenth International Symposium on Hardware/Software Codesign, CODES 2002.*, 2002, pp. 67--72.

[111] G. Ascia, V. Catania, and M. Palesi, "Parameterised system design based on genetic algorithms," in *Proceedings of the Ninth International Symposium on Hardware/Software Codesign, CODES 2001.*, 2001, pp. 177--182.

[112] T. S. Kumar, C. P. Ravikumar, and R. Govindarajan, "MAX: A Multi Objective Memory Architecture eXploration Framework for Embedded Systems-on-Chip," *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID07)*, pp. 527-533, 2007.

[113] E. Dahlman, S. Parkvall, J. Sköld, and P. Beming, *3G Evolution: HSPA and LTE for Mobile Broadband*, 2nd ed.: Academic Press, 2007.

[114] J. Berkmann, C. Carbonelli, F. Dietrich, C. Drewes, and W. Xu, "On 3G LTE Terminal Implementation - Standard, Algorithms, Complexities and Challenges," *2008 International Wireless Communications and Mobile Computing Conference*, pp. 970-975, 2008.

Included Papers

Paper I

Evaluation of SIMD Architecture Enhancement in Embedded Processors for MPEG-4 [1]

Abstract. *This paper presents our studies on the effects of using SIMD processor extension developed to enhance the processor performance for streaming applications. Our approach was evaluated using MPEG-4 encoding application as benchmark. Although MPEG-4 consists of many different operations, we concentrated on the sum of absolute differences (SAD), a major part of the motion estimation. The SAD was chosen because it is one of the most frequently used operations in MPEG-4 encoding. It is estimated to consume between 40%-80% of the total video encoding time when implemented on a general purpose processor. We have performed an extensive evaluation of our architecture extension. This evaluation showed that it is possible to achieve high performance with acceptable power consumption. We obtained about two times performance improvement for MPEG-4 encoding with roughly the same power consumption.*

[1] This paper is a reformatted version of *Evaluation of SIMD Architecture Enhancement In Embedded Processors for MPEG-4*, in Proc. Symposium on Digital Systems Design (DSD-04), Rennes, France, August 31 - September 3, 2004.

1.1 Introduction

The increasing usage of computationally intensive multimedia applications in mobile phones and personal data assistants (PDAs) puts new requirements on performance, power consumption and memory. Research on how to deal with such demands has mainly concentrated on desktop processors [1]. The research on embedded processors has been scarcer, with the result that processor architectural developments for low-power devices have been lagging behind. This in turn has been reinforced even more by the fact that most solutions in low-power embedded designs have been hardwired to cover shortcomings of most existing processors. While these solutions have had great impact, in the longer perspective they have had two main drawbacks. First, they do not provide flexibility, which is essential if one solution is to be able to be programmable for different standards and applications. Second, the process technology development introduces new design problems, which makes design of hardware even more difficult and time consuming. Therefore it reinforces even more the need for more flexible solutions. Software-based solutions, on the other hand, provide necessary flexibility to architecture but have to be carefully designed to provide sufficient processing performance, and still keep power dissipation low.

In this paper, we study the effects of using Single Instruction Multiple Data (SIMD) extension architecture to enhance the processor performance for streaming applications while providing the flexibility of a software solution. The media application we have decided as the benchmark is MPEG-4 encoding. This application consists of different operations and we concentrated mainly on the Sum of Absolute Differences (SAD), which is a major part of the motion estimation. The choice of SAD operation as a main metric for evaluation of the architecture is based on the fact that it is one of the most frequently used operations in MPEG-4 encoding. This computation consumes a significant portion of the total processing time. It is estimated to consume between 40%-80% of the total video encoding time when implemented on a general purpose processor [2].

The main evaluation of the architecture is carried out by a complete video encoding package. This is a full encoding application package developed by Ericsson for encoding and decoding of streamed media. It is based on state-of-the-art algorithms used in media applications. The importance of having a real application with realistic workloads is one key component when carrying out our architectural studies. Lacking this realism results often in an architectural design that has no actual value.

Since many multimedia applications work on streams of data, performing the same computation, SIMD architecture has been chosen as the main enhancement for an embedded processor. Our goal is to improve performance for critical data streaming applications and still keep the power under control. This is achieved by extending a general purpose embedded processor to handle heavy real time image processing. We have carried out an extensive evaluation to study different impacts of our extensions on

the entire system. This was done on real applications.

The interaction between memory and processor has a big impact on architecture performance due to a difference in processor speed and memory access time. This can often create a bottleneck in architecture and need to be considered during analysis. In our experiments, we have simulated memory and cache traffic to analyze its impact on performance.

We have embedded SIMD unit into MIPS processor by adding extended instructions to the Instruction Set Architecture (ISA). This approach utilizes the parallelism existing in streamed multimedia applications. For this study, we consider a specialized SIMD extension but it can be later expanded with other functionalities and instructions to support other vector operations, such as multiplication accumulation (MAC).

The structure of the paper is as follows. In Section 2 some background is given for media applications, especially MPEG-4. Section 3 describes the baseline architecture of the SIMD extension. Section 4 describes the method we used in our approach. Section 5 presents the experimental results. In Section 6 we discuss related work and in Section 7 we give some concluding remarks and future work.

1.2 Media applications impact on embedded architectures

The motion standards MPEG-1, MPEG-2 and MPEG-4 were developed in response to demands for higher data rates, increased efficiency in data coding and greater input-format flexibility [3]. As a consequence the standard has become complex with high requirements on implementation performance. The MPEG-4 standards define four different profiles of which we focus on the main profile, video, and in particular the encoding of moving images.

One important reason for developing MPEG-4 is that, unlike MPEG-2, it is object based. Video objects (VO), as they are called, are visual objects represented by the shape, motion and texture information. Each VO is comprised of Video object plane (VOP), which contains motion parameters. The VOP is encoded into a separate video object layer. It uses a window, where each object is divided into a grid containing a set of blocks, called Macroblocks (MB). The size of VOP is either 16 x 16 or 8 x 8 MB. Each MB consists of 16 x16 pixels. For each MB of the current frame, the block-matching algorithm uses a search window in a previously reconstructed frame. It searches for the closest matching block to the current block and defines a motion vector (MV) pointing to it.

There are different measures for finding the block with the best matching criteria. The most common one is sum of absolute differences (SAD). This operation is basically comparison between two consecutive MBs. It determines the amount of differences between these MBs. This operation makes the motion estimation (ME),

one of the most data-dominated and the most frequently used part of MPEG-4 application. The huge number of memory access put tremendous demands on the whole system. This is a reason why we have concentrated to propose and evaluate a specific SIMD extension for this operation. We address performance and power considerations as well as integration with the rest of the processor, especially memory system.

Given the growing importance of different media applications for future desktop and embedded processors, it is not surprising that there has been a lot of research in this field. The research can be divided in two categories; high performance and low-power.

In video encoding area most research focus has been on high performance computing, since this area includes many heavy media processing applications. The solutions that exist in this area are usually different kinds of ISA extensions, such as SIMD enhancement of general purpose processors. Video coding algorithms often process byte-wide data therefore a 32 or 64-bit wide arithmetic unit is underused. To make it more efficient a subword parallelism, in which a standard unit, is divided into smaller units has been proposed. These can be organized in different sizes but the most commonly used sizes are 8 or 16-bits [4].

In the low-power embedded domain the solutions that has so far been proposed and implemented have been ASIC centric, meaning design and development of a specific chip for each platform. A reason for this has been the lack of high performance solutions among embedded processors. One drawback with these solutions is the lack of flexibility. Another is the cost of developing a hardwired solution for each design and standard. This cost is not just a unit cost but it includes also costs for longer time to market. Thus the need to use other solutions, where both the processing requirements and the flexibility of running diverse applications and standards are met, has risen.

The main drawback with high performance studies have been the lack of relevance when it comes to drawing right conclusions with regards to low-power embedded systems. Having this in mind, one can use some of the proposed ideas to investigate the possibility of using them for low-power embedded systems. Taking these facts together and bearing in mind that media applications, such as MPEG-4, used in low-power area have different characteristics than ones used in high performance area, there is a room for introducing new solutions. Therefore we feel that this is an area needing more attention. Our investigation tries to bridge this gap by addressing high-performance, low-power and flexible solutions for mobile MPEG-4 applications.

1.3 Baseline architecture

Our proposal is to implement SIMD architecture with media extensions to handle streamed video applications, such as MPEG-4, for embedded platforms. These video applications are quite suitable for SIMD based solutions, as they often apply the same computation on an entire stream of data [5, 6]. In addition, SIMD design is highly efficient in exploiting the structure and resources of the processor. By using SIMD

extension and exploring both subword parallelism and streaming behavior of the MPEG-4 application, we want to enhance the overall performance so much that a soft solution is possible for the embedded domain.

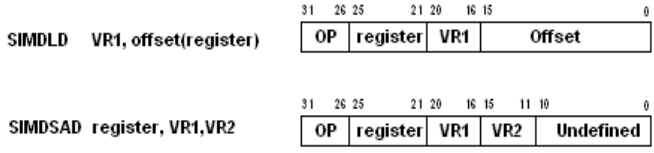


Figure 1. New SIMD instructions are 32-bit and follow the same bit structure as the other processor instructions.

The SIMD unit architecture is quite general and can be added to several embedded processors. In our studies, we have focused on MIPS processors and extended it with our SIMD unit for data streaming applications. The unit was integrated into MIPS processor by defining two new SIMD instructions. In figure 1 SIMDLD and SIMDSAD instructions are shown. These new instructions follow the same ISA as the other processor instructions. SIMDLD is a load instruction, which performs the loading of vector registers (VR1 or VR2). SIMDSAD instruction performs the SAD operation by taking VR1 and VR2 as input operands and writing the result, sum of the absolute difference, in the target register.

Figure 2 shows SIMD extension proposal together with the overall architecture. SIMD unit is integrated in the normal flow of the instruction pipeline of the processor. At the instruction decode stage SIMD instruction is identified and redirected to SIMD unit. Depending on the instruction, SIMDLD or SIMDSAD, it is executed differently.

The SIMDLD results in loading SIMD-vector registers VR1 or VR2. Each vector register consists of 16 8-byte registers. The bandwidth of the connection between the data cache and the SIMD unit is 64-bits, thus resulting in two or three accesses for loading each vector register. The three accesses are needed for alignment of data. Alignment is performed on the current frame macroblock. This is done by reading three 8-bytes from memory. These 24 bytes are then shifted left so that they become byte aligned, i.e., we get two 8-byte aligned data.

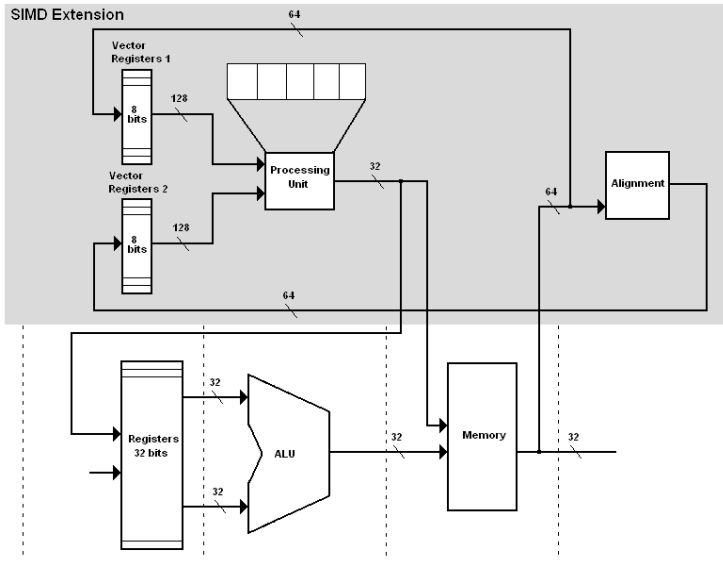


Figure 2. The proposed SIMD architecture. The connection from memory to SIMD unit is 64-bit wide.

SIMDSAD instruction operates on two vector registers, performing in parallel a sum of absolute differences. Basically, we have first 16 functional units, which perform absolute value operations in parallel for all pairs of data in vector registers. Then a tree of adders (together 15) makes addition of all these values. The vector processing unit is pipelined and has several stages. The number of stages depends on used adders and technology.

The choice of the adder is an important issue and could benefit or hamper the performance of the processing. According to 0.13μm UMC technology process an overall execution time for a one-bit full adder cell is 0.19 ns [7]. An 8-bit carry-ripple adder has a latency of 1.52 ns. Therefore, a pipeline of five stages can be run at 600 MHz. This would give us sufficient performance assumed in our experiments. Obviously, selection of more complex adder can result in lower number of stages or higher clock frequency.

The SIMD unit can be compared to a multi staged floating point unit, which is located often in parallel with the processors own Arithmetic Logic Unit (ALU). Result of SAD operation is a single value, which is placed back in one of the processor registers for use by the video encoding application. Processing unit together with the two vector registers and alignment unit is the core of the SIMD extension.

Introducing this extensive SIMD support affects many parts of the system, especially memory system and data path. This creates a need for investigating the architectural

design tradeoffs. One such tradeoff is how we can load the 32 SIMD vector registers with aligned data with as short as possible latency. This clearly introduces a classical bottleneck in the system.

1.4 Methodology

1.4.1 Simulation

To evaluate our approach we made simulations using SimpleScalar toolset [8]. This toolset provides an infrastructure for architectural modeling. The tool is open source, which enables a user to modifying it for particular architectural studies. It is also one of the most widely used toolsets for computer architecture studies. This is an execution-driven simulator, which in our case simulated an MIPS instruction set architecture. With the integration of the power estimation tool Wattch [9], the evaluation system for our baseline architecture was thus in place. This environment was used for our evaluation of the encoding application.

The power model used for the architecture has been executed for the SIMD instructions. The switching information for registers, functional units and buses is collected and used by Wattch for power calculation.

1.4.2 Application

The application that was used for the main evaluation was a complete video encoding package. This application is a full encoding application package developed by Ericsson for encoding and decoding of streamed media [10]. It is based on algorithms used in real media applications. Ericsson Mobile Platforms provided this application within the framework for cooperation in study of future mobile platforms.

For evaluation SIMD extension we chose one of the main operations in the MPEG-4 encoding, the Sum of Absolute Differences (SAD). This operation is done in motion estimation part of the digital video encoding. One important aspect of the SAD operation is the fact that it is often used for evaluation of different architectures. It is used because it is the most frequently executed operation in video encoding [5, 11]. Many other researchers have also chosen this operation in their studies [2, 5]. This is also why having an efficient solution for SAD is of great importance with regards to performance.

One of the most important issues when doing architectural studies is the accuracy of the applications used. Being able to use a real workload improves the overall architectural design. This issue is even more crucial for data intensive applications, such as MPEG encoding.

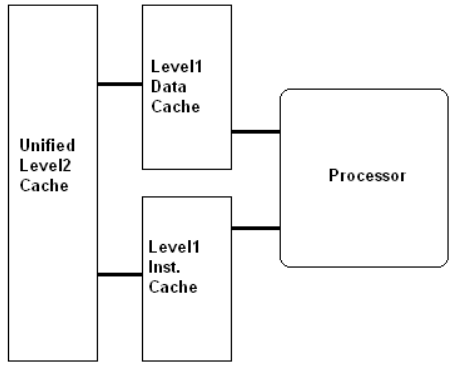


Figure 3. Modeled baseline architecture with separate data and instruction level1 caches together with a unified level2 cache.

1.4.3 Architecture

We start evaluation of our approach by examining an embedded processor with SIMD media extensions. The evaluation results of this architecture acted as basic reference for power and performance. The memory model used for evaluation was Harvard architecture with a unified second level cache, depicted in Figure 3. As our proposal was an embedded SIMD design the main focus was on the processing unit, CPU, instruction- and data-caches, memory and data paths. From this baseline architecture we added new functionality, by integrating a parallel data-path in the execution stage so that it handled SIMD instructions. The SIMD unit is directly connected to the data-cache, which provides the necessary bandwidth and direct access. Table 1 lists the cache configurations and table 2 shows the system configuration used in our experiments. For power consumption we assume 0.13μm UMC technology.

Table 1. Memory configuration of the modeled architecture.

	Size (KB)	Block size (Bytes)	Associativity	Replacement policy
L1 Data Cache	512	64/32	4	LRU
L1 Instruction Cache	512	32	1	LRU
Unified L2 Cache	1024	64	4	LRU

Table 2. System configuration and memory latency.

System Clock	600 MHz
Memory access bus width	16-bytes, 8-bytes
Memory access latency	18 cycles

1.5 Experiments and Discussion

In this section, we present the evaluation of our proposed architecture. First, we consider the kernel SAD loop, which is used for initial evaluation of our architecture. We also comment simulation results for this case. In subsection 2 we start by explaining the characteristics of MPEG-4 encoding application, its configuration and test sequences which were used. Next section provides results from different executions. The following section deals with the impact of the ISA extension. Finally evaluation of the results and their implications are discussed in subsection 4.

1.5.1 Kernel SAD loop

A test of the kernel code of SAD loop was written for initial evaluation of the SIMD extension. The kernel code is shown in figure 4 for both non-SIMD processor and processor with SIMD extensions. The main characteristic of the loop is the amount of potential parallelism that is present. The SAD operation works on macroblocks (MB) as discussed in section 2. This results in a loop containing 16 absolute difference operations, which at the end are added to a sum. The inner loop is thus replaced by the SIMD instructions. This loop is run 16 times since in our case MB is 16x16. The total sum of all absolute differences is the result of SAD calculation.

```
for (i=0; i<16; i++){
  for (j=0; j<16;j++){
    sad+=abs(b[i][j]-c[i][j]);
  }
}
```

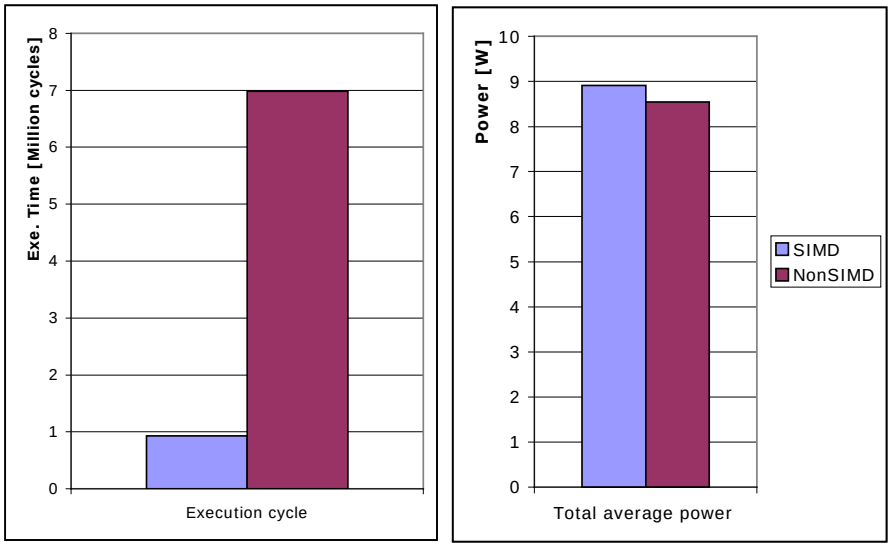
(a)

```
for (i=0; i<16; i++){
  R2=&b[i][0];
  R3=&c[i][0];
  SIMDLD VR1, 0(R2)
  SIMDLD VR2, 0(R3)
  SIMDSAD R4, VR1, VR2
}
```

(b)

Figure 4. SAD loop, (a) without SIMD extension and (b) with SIMD extension. In the case with SIMD extension an inline assembler code represents the SIMD instructions SIMDLD and SIMDSAD. R2, R3 and R4 in (b) are normal processors registers.

Simulations of the SAD kernel code produced performance and power consumption evaluation. These results verified the correctness of functionality of SIMD extension and provided performance and power consumption figures. Figure 5a shows the results obtained from running a SAD loop 1000 times. These results indicated to some degree a promising performance by providing a boost of over 7 times. In addition, as can be seen in figure 5b power usage is insignificantly higher with SIMD extended architecture. Since these experiments were run on SAD kernel code there are very limited conclusions that could be drawn from them, especially with regards to memory and overall performance. Therefore the need for complete application was required.



(a) (b)

Figure 5. (a) Execution cycles after running SAD loop. The diagram shows a performance increase of over 7 times for the SIMD extended architecture. (b) Shows the average power for SIMD and NonSIMD.

1.5.2 Configuration of MPEG-4 application

Table 3 shows the configuration of the application chosen for evaluation of the architecture. Resolution was set to QCIF 176x144. This is the size regularly used in handheld devises. Output bit or data rate was set at 384 kbit/s. This data rate is used by 3GPP, which is the standard for third generation mobile phones [12]. Test sequences chosen as input are called news and mobile. The main difference between them is the amount of movement in the sequence of video images.

Table 3. Configuration of MPEG-4 application.

Output Resolution	QCIF 176x144
Data rate	384 kbit/s
Advanced Intra Coding Mode	ON
Deblocking Filter in the Loop	ON
Modified quantization Mode	ON
Rate control	NRC

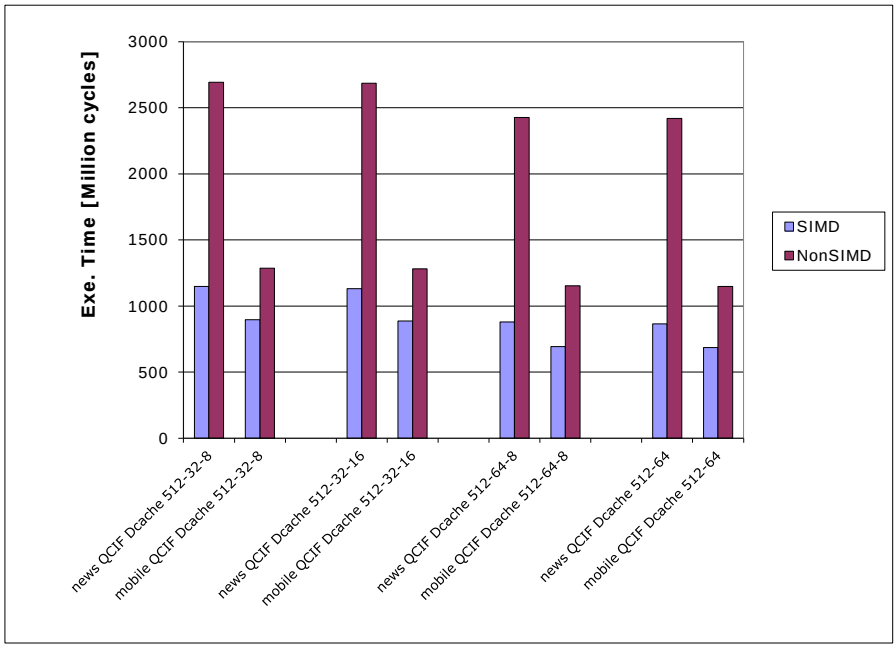


Figure 6. Execution cycles for running MPEG-4 application with two different test sequences news on the left and mobile on the right. The bars on left show the execution cycles for architecture with a 512 KB memory and block size 32 bytes. The right bars are with same cache size but with block size 64 bytes.

1.5.3 Impact of SIMD extension

After running a series of simulations the results show the potential for SIMD extension is quite significant. Figure 6 shows the reduction of the total execution cycles to almost half. The first major observation is the impact of input test vector where mobile sequence has significant lower execution cycles. The second observation is the impact

of different cache configurations; 64-bytes block size instead of 32-bytes and memory access bus width, 16 bytes instead of 8 bytes. As shown in all cases the performance of the SIMD architecture is almost two times better than the NonSIMD architecture.

We also observed a significant reduction in cache accesses both for instruction cache as well as data cache as seen in figure 7. The decrease of cache accesses for instruction cache is obvious as we have SIMD architecture. The reduction of data cache accesses comes from the fact that for each access in SIMD architecture, our extension loads 8 pixels (i.e., 64-bits). NonSIMD architecture, on the other hand, loads only one pixel (i.e., 8-bits). We can also see the difference between the two test sequences news on the left and mobile on the right. This clearly shows the influence of input data on the behavior of the architecture, as with the case for execution cycles above.

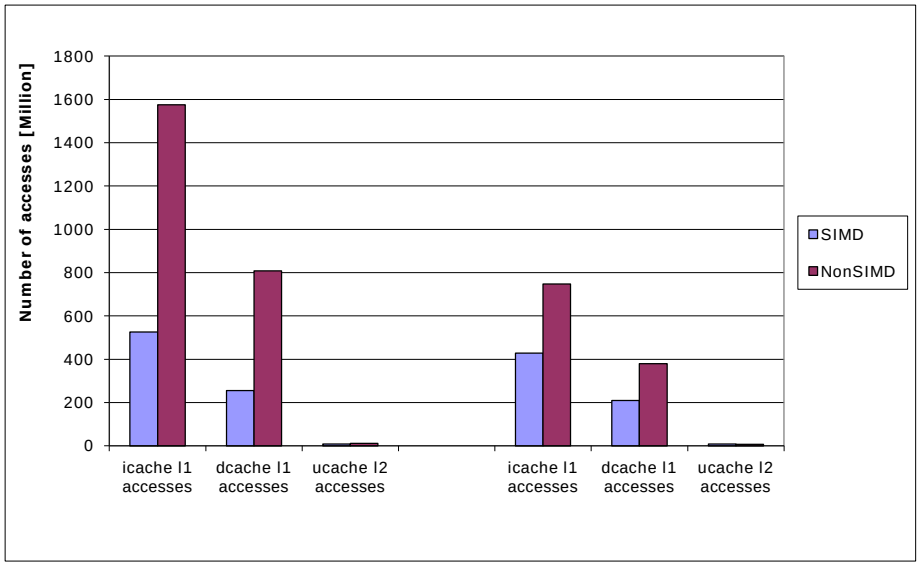


Figure 7. Cache accesses for SIMD and NonSIMD show a significant decrease both for instruction- and data level 1 cache. The diagram on the left is cache accesses for news test sequence. The diagram on the right is cache accesses for mobile test sequence.

The total power consumption is usually lower for SIMD even though we have introduced a new component in the processor architecture. Figure 8 shows the average power consumption for the SIMD unit and caches. This average is basically power consumption per cycle. The SIMD extended architecture also shows a lowering of the total average power. Data cache is the most contributing factor to the total power consumption figure. This lower power consumption can be explained by the fact that SIMD architecture has lower number of cache accesses (see figure 7). However, these

accesses are 64-bits instead of 32-bits. As can also be seen the average power of the SIMD unit is insignificant compared to the total average power. The data on the left is news test sequence and on the right mobile.

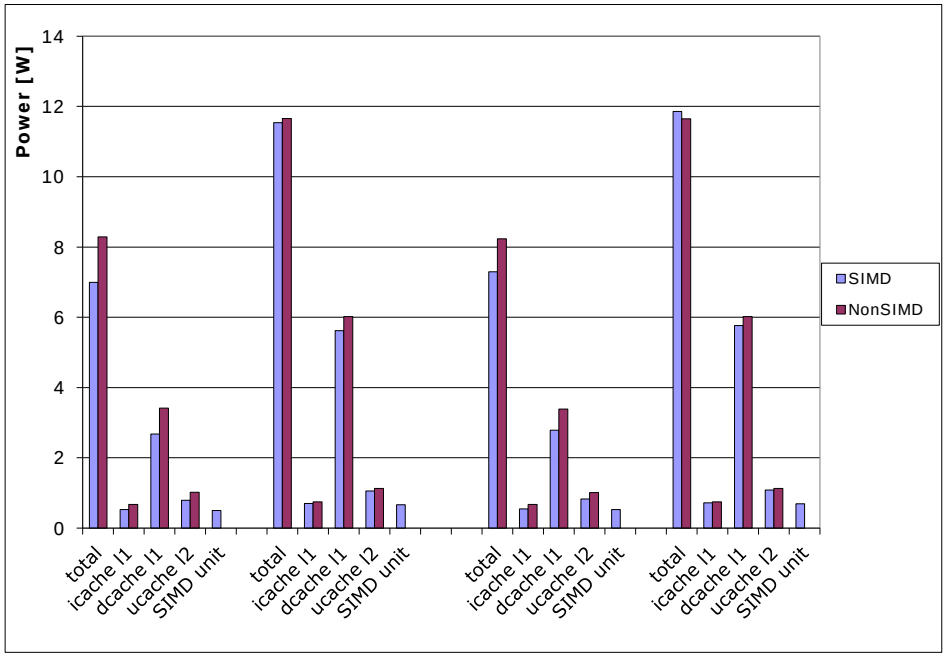


Figure 8. Power consumption for SIMD and NonSIMD architecture for test sequences news and mobile. The cache configuration chosen here was 512-32-8 and 512-64-16 from figure 6.

1.5.4 Main observations

Adding SIMD extension to an embedded processor for increasing the performance is one way of solving the shortcomings of today's embedded processors. One of the benefits with this solution is its flexibility. Instead of adding a hardwired solution beside the processor, tailored for a certain standard, we use flexibility and programmability of general purpose processors. The cost for this solution is small and insignificant with regards to power increase.

The most interesting observation is the increased importance of data paths. By increasing the performance of the processor, new problems arise on how to provide the processor with enough data so the increased processor performance is not lost. The results in the previous section clearly illustrate this. The increased cache size and its right configuration, together with increased memory bus width produce a significant improvement in the overall performance of the system.

Another important issue is the complexity of MPEG-4 encoding application that is seen clearly in figure 6. Execution of different test sequences, such as mobile and news, influences the overall system load. In the case of figure 7 we have quite high number of data accesses for news compared to mobile. Mobile test sequence, in fact, requires more computations but as the target data rate is not achieved the application lowers the output quality to meet the target data rate, thus reducing the number of computations. Figure 9 shows the performance changes of the system when changing the configuration of the application. The most striking factor is that the amount of computation increases as the data rate restrictions are relaxed. This increase is much more significant for the NonSIMD. This indicates that more time is spent on computing to achieve better image quality when higher data rate is available. Thus the impact of our SIMD extension is more dominant. This increase in available bandwidth is something that future wireless systems will provide. Thus having high data rates in excess of 768 kbits/s is realistic.

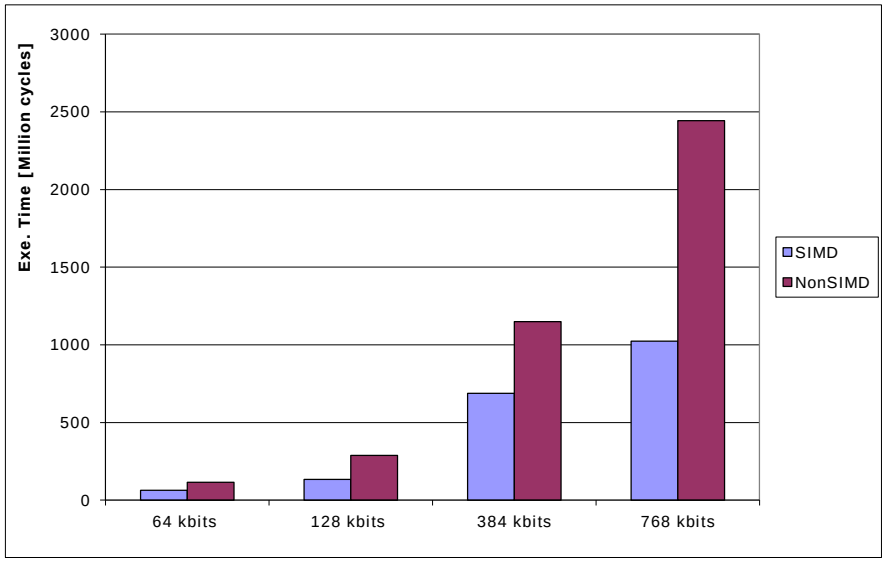


Figure 9. Execution time for mobile test sequence with different target data rates. The data rates chosen are from 3 GPP standard. We see an increase of execution time as we increase the target data rate.

To conclude, an increase in performance can only be justified as long as the rest of the system can also be improved by the same amount. Otherwise the added performance will not be justified. Thus improved data path from memory to processor is a must to move forward with any new improved embedded processors.

1.6 Related work

There has been some work done to address SIMD, which also takes into consideration the system impacts [4, 5, 13]. Lappalainen et al. give evaluation of different algorithmic optimizations for media applications. Their study was done using MultiMedia eXtensions (MMX) and Streamed SIMD Extensions (SSE) on Intel Pentium III [14]. This study does not discuss power consumption or silicon area and cost optimizations.

Many of the studies are based on desktop processor architectures. They have some similarities with low-power embedded processors but there are in many ways different, both with regards to architecture and performance. The sensitivity for silicon area budget as well as power budget is one point that seldom is investigated in high-performance research. Most studies, which consider low-power and high-performance have focused on hardware solutions [15] or Digital Signal Processors (DSPs) [16] as basic blocks in embedded systems for media applications.

The solutions, which were presented before for desktop processors are now appearing in low-power embedded processors, as pointed out in [5]. This development is also observed with leading low-power embedded processor manufactures, such as ARM, MIPS and Hitachi. In their new processors a limited form of high-performance SIMD architectures has been added [17, 18]. This indicates that a processor implementation is possible, which in turn introduces some new aspects for media applications. Thus opening the door for more radical implementations using the flexibility that these solutions provide contra hardware based solution or even a DSP based solution. An ideal SIMD solution should have none or very minor impact on the whole system. But as it is commonly known, introducing new design aspects into the system without fully evaluating the system impacts have in reality very little value. The impact on memory and cache is of particular interest to study, as these system structures are of even more importance in embedded systems.

One of the implementations that consider many of the embedded low-power design constrains such as silicon area and power dissipation is ARM processors. In order to give clear understanding of our approach compared to ARMv6 a more detailed description is needed. ARMv6 is the latest processor architecture that ARM has produced. Among the newly added features is a limited embedded SIMD capability. It uses a 32-bit Arithmetic Logic Unit (ALU) and a 64-bits bus which provides performance improvement between 2 and 4 times [18]. This clearly shows that adding SIMD to processor core has great potentials for improving performance, and still keeping embedded design constrains. But these performance gains are not close enough for replacing a dedicated hardware for video encoding. Instead a more extensive SIMD solution is needed, to be able to meet the computational requirements from the encoding application.

1.7 Conclusions and future work

We have proposed and evaluated SIMD architecture for MPEG-4 encoding for enhancing the performance of embedded processors. We have shown in our study that by using SIMD extension we can increase the overall performance of an embedded processor and still keep the power at similar level as original architecture. These results are clearly related to the issue of memory bottleneck, one of the most important problems when dealing with embedded systems. As the margins for embedded systems are much narrower than high performance systems, the system imbalances cannot be neglected in the same way. Thus we see a need for more in-depth work around memory and processor interaction. Our future work will concentrate on SIMD extensions with a focus on the memory issues, where we see a gap in solutions for embedded systems that can provide the necessary high data bandwidth. We plan also to improve the overall performance of the SIMD extension execution part.

References

[1] S. A. McKee, Z. Fang, M. Valero, "An MPEG-4 Performance Study for non-SIMD, General Purpose Architectures" in Pros. ISPASS. 2003 IEEE International Symposium, 2003, pp. 49-57.

[2] D. Guevorkian, A. Launiainen, P. Liuha and V. Lappalainen, "Architecture for the Sum of Absolute Differences Operation", in Proc. 2002 IEEE Signal Processing Systems, SIPS '02, pp. 57 –62.

[3] MPEG-4 Final Draft International Standard 144 496-2 ISO/IEC JTC1SC29/WG11/N2502.

[4] M. Ferretti, "Multi-media extensions in super-pipelined microarchitectures. A new case for SIMD processing?" in Proc. 5th IEEE Int. Workshop Computer Architectures for Machine Perception, 2000, pp. 249-258.

[5] V. Lappalainen, T. D. Hämäläinen and Petri Liuha, "Overview of Research Efforts on Media ISA Extensions and Their Usage in Video Coding" IEEE Trans. Circuit and System for Video tech., vol. 12, no. 8, Aug. 2002.

[6] S. Vassiliadis, B. Juurlink and E. Hakkennes, "Complex Streamed Instructions: Introduction and Initial Evaluation" in Proc. 26th Euromicro Conference, 2000, vol.1, pp. 400-408.

[7] Virtual Silicon 0.13um High Density Standard Cell Library, United Microelectronic Company, UMC process.

[8] T. Austin et al., "SimpleScalar: an infrastructure for computer system modeling", IEEE Computer, Vol. 35, Issue 2 , Feb 2002, pp.59-67

[9] D. Brooks, V.Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In International Symposium on Computer Architecture, pages 83-94, June 2000.

[10] VIPS video package developed at Ericsson Compression Lab (CLAB).

[11] P. Kuhn, “Algorithms, Complexity Analysis and VLSI Architecture for MPEG-4 Motion Estimation”, Kluwer Academic Publishers, 2003, ISBN 0-7923-8516-0

[12] 3 Gpp standard, http://www.3gpp.org

[13] S. M. Akramullah, Software-based video encoding using high-performance computing,” Ph.D. dissertation, Dept. Elect. Electron. Eng., The Hong Kong Univ. of Sci. and technol., Hong Kong, 1999.

[14] V. Lappalainen and T. D. Hämäläinen, “Unified Method for Optimization of Several Video Coding Algorithms on General-Purpose Processors” in Proc IEEE Int. Conf. on Information Technology: Coding and Computing, ITCC 2002

[15] T. Kamemaru et al., “Media processor core architecture for realtime, bi-directional MPEG4/H.26X codec with 30 fr/s for CIF-video”, in Proc. IEEE 2000 Custom Integrated Circuits Conference,. CICC 2000, pp.473-476

[16] M. Berekovic, R. Frase and P. Pisch, “A Flexible Processor Architecture for MPEG-4 Image Compositing” in Proc. 1998 IEEE Int. Conf. Acoustics, Speech, and Signal Processing, ICASSP '98, Vol.5, 12-15 May 1998, pp. 3153 -3156

[17] D. Cormie, “The ARM11 Microarchitecture”, ARM Limited, Cambridge, UK, Apr 2002, http://www.arm.com

[18] D. Brash, The ARM Architecture Version 6 (ARMv6), ARM Limited, Cambridge, UK, Jan 2002, http://www.arm.com.

Paper II

Analysis of Embedded Processors for Streaming Media Applications[2]

Abstract. *Streaming media applications, such as MPEG-4, impose great challenges on embedded systems. In this paper, we present our analysis of an embedded processor family used in most of wireless handheld devices. For our analysis and evaluation purpose we use full video MPEG-4 encoding application with optimized algorithms for mobile devices. We observed significant performance improvements when using new architectural solutions in embedded processors. There are two sources of this improvement. The first one is due to introduction of Single Instruction Multiple Data (SIMD) extension in embedded processors. The second one comes from increased clock frequency enabled by deeper instruction pipeline. These architectural enhancements give an increased performance, but are not sufficient in providing enough performance for heavy video encoding (CIF screen size images at 30 frames per second). For achieving this, a more aggressive SIMD unit is required as indicated by our proposed architecture and its analysis.*

[2] This paper is a reformatted version of *Analysis of Embedded Processors for Streaming Media Applications*, in Proc. of the 8th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW-8), San Francisco, USA, February. 12, 2005.

2.1 Introduction

The evolution of embedded processors has been a steady progress from simple micro-controllers, to today's advanced multi-staged pipelined designs. This evolution has come mainly from the demands applications have put on embedded systems. These applications differ not only in complexity but also in the amount of processing load they require. A typical streaming media application, video encoding, involves discrete cosine transform as well as on-the fly interpolation of pixel values during search in motion estimation. These demands often require highly parallel solutions to achieve even mildly reasonable results.

MPEG-4 video encoding on mobile handheld devices is a good example of a heavy application running on a resource limited device. Mobile handheld devices have narrower design constrains, such as limited energy supply and limited size. They have become hard tuned and designed with very confined demands. With this in mind, these devices have different design challenges than high-performance designs. The key issue has been to increase the performance just enough to solve the problem with little cost overhead. But there is a new issue involved here, flexibility. Flexibility is important in future system architectures, as the need to be able to use the same hardware for different applications and standards has risen. Therefore, an important issue in embedded processor design has been efficiency, in terms of gained performance at a cost of increased design complexity and resource utilization.

The trend of embedded processors evolution has taken two paths, one going to multi-processors and the other extending processor architecture with specific hardware to boost the performance [1]. In this paper, we concentrate on the second solution and analyze a few processor extensions needed for streaming media applications.

The objective of our research is to investigate and propose new architectures to increase the performance of handheld mobile device, by introducing specific hardware in the processor core. The main idea is to extend the Instruction Set Architecture (ISA) of a processor with Single Instruction Multiple Data (SIMD) instructions. The choice of a SIMD design is natural, as media applications are parallel and well suited for vector units [2]. In our preliminary work we looked at the performance increase of the most critical parts of video encoding application through profiling. The operation that uses most of execution time is the Sum of Absolute Differences (SAD) in the motion estimation block. The calculation of the motion vector accounts for roughly 40-80% of total execution time [3]. Therefore we would like to analyze the impact on performance, when specific critical parts are boosted.

In this study we want to elaborate and analyze how some of today's embedded processors extensions can be used for streaming media applications. For our study we chose to look at the ARM processor family, as ARM is one the leading manufactures of

embedded processors. ARM926EJ (implements ARMv5 ISA) was chosen as our baseline processor that is one of the most widely used designs in mobile embedded devices. As ARM1136EJ (implements ARMv6 ISA) is the successor of ARM926EJ, it was chosen to represent the next generation of processors. Among added features ARM1136EJ has the new ARMv6 ISA that was introduced by ARM to provide SIMD capability with data width of 32 bits. The idea of adding signal-processing horsepower to general-purpose processors is natural and is used in most processors [4]. Embedded processors follow this path as well, which is also the direction ARM has taken. The advantage of such approach is that it does not increase the area required by the CPU that much [5]. It can eliminate the need of additional DSP core or specific hardwired logic [1].

The idea of SIMD instructions for multimedia applications has been widely used. In most cases the main evaluations of architectures have been done by using different sets of kernels [6]. This is certainly a good approach in the introductory phase of evaluation. But in order to do in-depth analysis and evaluate overall speed-up of architecture, one needs to use complete applications [7]. There is also the issue of the quality or relevance of benchmarking architecture with a reference application [8]. The reference application has often not been optimized and, thus, behaves differently from commercial applications. Another issue affecting embedded processor research is that most evaluations are carried out on high-performance processors such as Pentium or PowerPC. This led to conclusions that cannot be automatically transferred to embedded processors [9].

Our evaluation was done using ARM's instruction set simulator ARMulator in Realview tool set. This is a cycle accurate simulator which is used for software as well as hardware development. Our analysis of the architectures was done using a full MPEG-4 encoding application. This application was provided to us by Ericsson Mobile Platforms AB and uses realistic optimized algorithms for mobile devices.

Our results show that the road we have taken by extending an embedded processor with specialized SIMD extension is the right way forward [10], both with regards to gained performance as well as power consumption. What differs our approach from ARMv6 extension is the extent of how massive in terms of data width this SIMD unit should be. In the case of ARM, the unit is 32-bits, indicating that it can initiate 4 parallel operations using one instruction. In our own architecture we look at data width of 128-bits and beyond.

ARM is not the only embedded processor design that is moving in the direction of extending the ISA with SIMD capability. MIPS technologies is also starting to enhance their processors with SIMD units by introducing new SIMD instructions in their MIPS IV and MIPS V ISA [11].

The structure of the paper is as follows. In Section 2 basic information on ARM architecture is given. Section 3 describes our video encoding application and in section 4 the method we used in our approach. Section 5 presents our experimental results. In

Section 6, we discuss experimental results and in Section 7 we give some concluding remarks.

2.2 ARM Architecture

The next generation of embedded processor architectures has been driven by the needs of emerging products, which require more processing power. The key design constrains in the embedded domain have been performance, power, area, and cost. These factors must be balanced to meet the requirements of each application.

One of key areas the new ARMv6 architecture focuses on is multimedia application. Single Instruction Multiple Data (SIMD) capabilities enable more efficient software implementation of these applications. These new instructions provide performance improvements between 2x to 4x, depending on the multimedia applications [12]. ARMv6 is based on a 32-bit processor, the same as ARMv5, which is implemented in ARM926EJ, ARM10 and XScale. Combined with a 64-bit bus support, this provides equivalent to a 64-bit machine, but without the power and area overhead of full 64-bit CPU [12]. The ARM1136EJ 64-bit data paths allow two instructions to be fetched from the cache in a single cycle, thus achieving high performance on code sequences where data is required to be moved in parallel with processing.

The ARM architecture is a load-store architecture, where the ARM core instructions can only operate on data in registers. Load and store instructions are used to transfer data to and from this register file. The L1 memory system has no wait states and runs synchronized to the core. The cache in our experiments is organized as a Harvard architecture with separate instruction and data caches.

The ARM media extensions are implemented for the first time in the ARMv6 designs. They include a set of SIMD instructions, as well as Sum-of-Absolute-Differences (SAD) support. The new instructions support 8 and 16 bit SIMD arithmetic, including four 8-bit and two 16-bit operations, parallel add and subtract, selection, packing and unpacking [12]. ARMv6 provides support for SAD calculation, with the inclusion of USAD8 (sum of differences) and USADA8 (sum of differences accumulate) instructions. Table 1 shows the relative performance of SAD operation, comparing ARMv6 and ARMv5 [12].

Table 1. Implementing Sum of Absolute Differences on the ARMv5 and ARMv6 architecture [12].

Architecture	Cycles/4 pixels
ARMv5	18
ARMv6	3

The target frequency range for ARMv6 is 500-700Mhz. To deliver a good performance power ratio the ARMv6 uses both clock frequency and supply voltage scaling. Enabling the system designer to control power consumption and performance. ARM1136EJ consumes less than 0.8 mW/MHz in a 0.13µm process technology. The typical ARM1136EJ synthesized core without caches takes 2.85 mm2 [13]. Compare with 2.2 mm2 for ARM926EJ in the same 0.13µm process technology with 0.5 mW/MHz [13].

The ARM1136EJ has a single-issue pipelined microarchitecture, which differs from previous ARM cores. It consists of 8 stages compared to 5 stages in ARM926EJ. This enables to increase the throughput by as much as 40% [14]. An obvious drawback with deep-pipelined structures is introduction of excessive latencies into the system. These latencies occur because of existing dependencies between instructions. Another issue for long pipeline processors is how to smooth program flow during branches. ARM1136EJ partially avoids these delays by using dynamic and static branch prediction to predict the flow of instructions. These two techniques are used to maintain good pipeline efficiency through removal of stalls. The result is the same effective latency as a 5-stage ARM926EJ but with higher throughput [12].

The two techniques used for predicting branches are dynamic branch prediction using a 4-state branch target address cache. It holds historical record to see whether the branch has been seen before, and whether it was most frequently taken, or most frequently not taken. If the dynamic branch predictor cannot find a record of the branch instruction, a static branch prediction takes over. Depending on the target address, it is going backwards or forwards. If it goes backwards, then branch is taken as it assumes it is a loop. If it is forward branch then it is not taken. Analyzing results from benchmark suites show correctness of the static branch predictor to around 77% of the time. When using only dynamic branch predictor, the processor correctly predicts 88% of the branches. When the effect of both static and dynamic branch prediction is combined, 92% of the branches are correctly predicted [14].

Although the front-end of the pipeline architecture is single issue, the back-end exploits the parallelism with separate processing units for the ALU, multiply-accumulate (MAC) and Load-Store (LS) instructions. This parallel structure enables the processor to proceed with other ALU and MAC operation when the LS unit has been stalled. This means an out-of-order completion.

The longer pipeline of ARMv6 should give about 35% clock boost over ARMv5 architectures. Theoretically then a hand tuned version of the core could be run at 1.6 GHz compared to the 1.2 GHz for ARM1020 with ARMv5 architecture [15]. The typical clock frequency for a 0.13µm process is around 500-700 MHz for ARMv6 architecture [12, 14, 15].

In our experiments, we want to evaluate this architecture and analyze the impact it has on performance, when heavy media streaming applications with realistic loads are benchmarked. We selected a special version of MPEG-4 application implemented to be

executed on handheld devices, which was also optimized for this purpose.

2.3 MPEG-4 Application

The main media application used in this study is a full video encoding MPEG-4 application. One important reason for developing MPEG-4 is that, unlike MPEG-2, it is object based. Video objects (VO), as they are called, are visual objects represented by the shape, motion and texture information. Each VO is composed of Video object plane (VOP), which contains motion parameters. The VOP is encoded into a separate video object layer. During encoding the MPEG-4 application uses a window, where each object is divided into a grid containing a set of blocks, called Macroblocks (MB). The size of VOP is either 16 x 16 or 8 x 8 MB Each MB consists of 16 x 16 pixels. For each MB of the current frame, the block-matching algorithm uses a search window in a previously reconstructed frame. It searches for the closest matching block to the current block and defines a motion vector (MV) pointing to it.

There are different measures for finding the block with the best matching criteria. The most common one is sum of absolute differences (SAD). This operation is basically a comparison between two consecutive MBs. It determines the amount of differences between these MBs. This operation makes the motion estimation (ME), one of the most data dominated and the most frequently used part of MPEG-4 application. This application has been provided to us by Ericsson Mobile Platforms AB. It features *full search* and *optimized search* modes, with the later being a more realistic algorithm for mobile devices. The optimized search mode has a number of features, such as early termination and improved search algorithm.

The application was configured for running screen size Common Image Format (CIF) (352x288) at 30 frames per second. We used two sets of algorithms. One based on full search with standard 8-point half pixel motion estimation, with search window of 17x17 macro blocks. The other algorithm is optimized commercial motion estimation with the search window of 31x31 macro blocks. The profiling of the application shows that the SAD at least accounts for 25-37% of the total execution time depending on the search algorithm used and test sequence characteristics. The case that uses full search with the CIF screen size at 30 frames per second is the upper bound for number of operations per second that are required for encoding.

2.4 Methodology

In our experiments we use three different test sequences, **mobile**, **foreman** and **news** [16]. The main difference between these sequences is the amount of processing they need to encode the sequences. Depending on the configuration of the application the test sequences make the application behave differently. The **mobile** sequence is the most demanding sequence in terms of processing when rate control restriction is

removed. This removal of the rate constraint is crucial when studying the architecture and not the application, as we need a stable behavior of the application. The impact of this restriction on the quality of the encoded video sequence is substantial. It is the rate control that makes the application skip frames when not being able to achieve the required target bit rate. In order to remove this uncertainty we relaxed the bit rate restriction to ensure that all frames are encoded and nothing is skipped due to rate control.

Another quality limiting factor is the quantization in the Discrete Cosine Transform (DCT), as this is the source of quality loss in the coding. We chose here a fix quantization to have a stable behavior of application when performing the experiments. The choice of a reasonable quantizer value has an important impact on the encoded video sequence. Choosing a too high quantizer will reduce the total bit rate but will also lower the quality of the encoded video sequence. Choosing a too low value will result in unreasonably high bit rate. We chose a quantizer value of 15 both for P frames and I frames, in order to have a reasonable quality of encoded video sequences.

First we ran the kernel code, in order to be able to determine the performance of both the processors as well as the impact of SIMD instructions. The kernel SAD code was run on ARM926EJ and ARM1136EJ. In the case of ARM1136EJ, for evaluating the SIMD instructions, inline ARM SIMD assembler instructions were used. We use two measures to evaluate the performance of the architectures. The first one is the number of cycles needed to execute the application (see figures 1-3). The second one measures the encoding performance of the architecture counting number of frames per second encoded (see figures 4-5).

The ARM instruction set simulator ARMulator was used for our evaluation. This is a cycle accurate simulator, which is part of the Realview developer's suite [17]. The memory architecture we used for our experiments was Harvard architecture with 32K instruction and data caches. The higher level memory was modeled as zero-wait. This is a too optimistic assumption, but when compared to our architecture with same memory hierarchy the impact of a level2 cache and memory contributes negligibly, around 3%, to the overall cycle counts.

To be able to compare ARM's SIMD extension with more aggressive SIMD unit we used our own SIMD extension [10]. Our proposed architecture was a 128-bits SIMD extension integrated in an embedded processor, thus extending the instruction set architecture. The memory model used for evaluation was also Harvard architecture with a unified second level cache. Since our proposal was an embedded SIMD design, the main focus was on the processing unit, CPU, instruction- and data-caches, memory, and data paths. To this baseline architecture we added new functionality, by integrating a parallel data-path in the execution stage so that it handled SIMD instructions. The SIMD unit is directly connected to the data-cache, which provides the necessary bandwidth and direct access. The ARM1136EJ was simulated to run at 600 MHz and the ARM926EJ at 300 MHz.

2.5 Experimental Results

The first aim of our experiments is to see how much improvement we have between the ARM processors with the same media application. The second is to compare performance improvement between ARM1136EJ with and without SIMD extension. In all cases we use the same application with the same configuration.

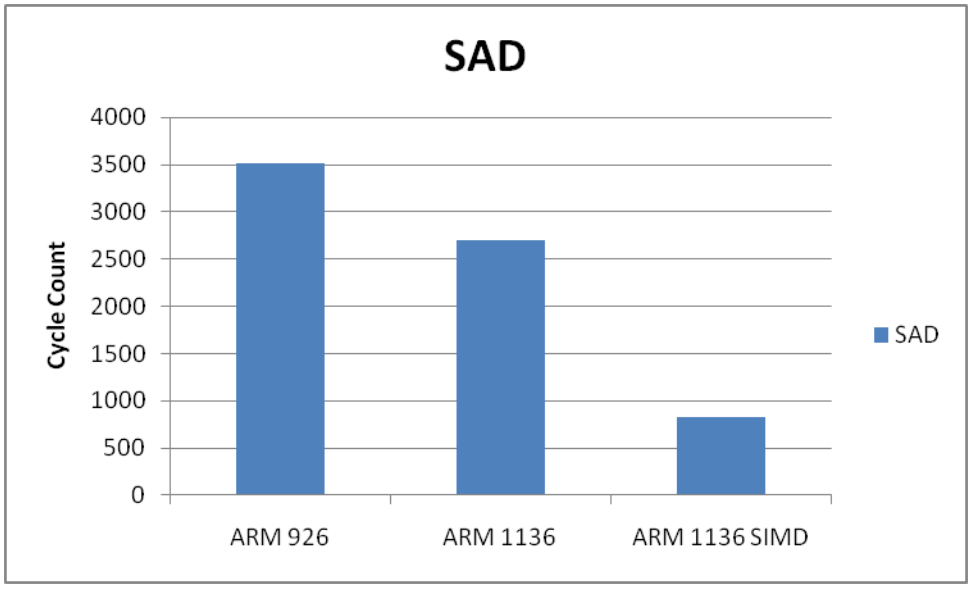


Figure 1. Cycle count for running Sum-of absolute differences operation using ARM 926EJ and ARM1136EJ without and with SIMD SAD instructions.

To evaluate the performance of the SIMD instructions we wrote a test kernel to determine the amount of maximum improvement we could get for SAD. Figure 1 shows the performance in cycles for ARM926EJ and ARM1136EJ without and with SIMD instructions. We observe performance improvements of about 3 times between ARM1136 without and with SIMD instructions. The difference between ARM926EJ and ARM1136EJ with SIMD is about 4.2 times. This falls short from the 6 times presented by ARM in [12] (table 1 section 2). Figure 2 shows the cycle count for running three different sets of test data sequences: mobile, foreman and news presented in section 3. The cycle count drops when moving from ARM926EJ to ARM1136EJ even though the architecture of the ARM1136EJ has 3 pipeline stages more than ARM926EJ. This increase in pipeline stages is one way of improving the architecture to get the benefit of running the processor core on a higher clock frequency. When this benefit is taken into account the performance impact is even more

significant. The ARM926EJ core is run at 300 MHz and the ARM1136EJ core is run at 600 MHz.

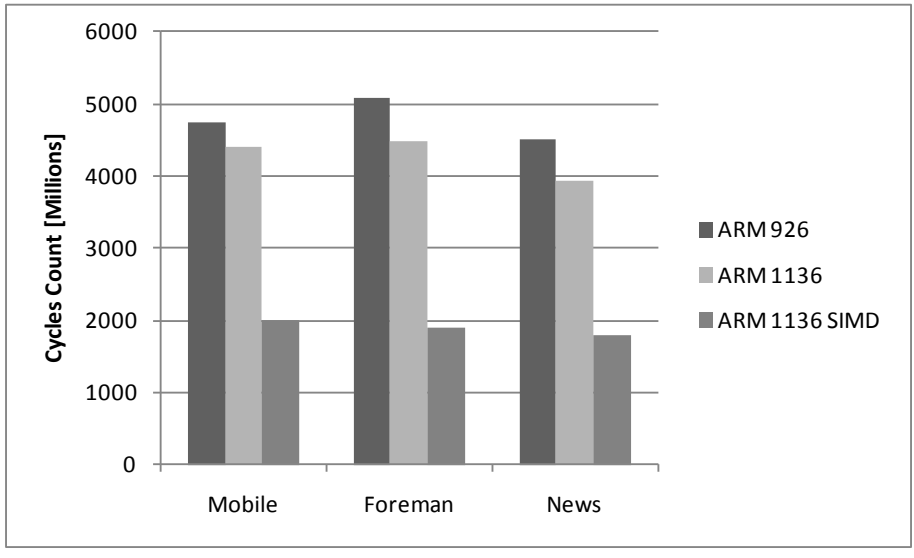


Figure 2. Cycle count for running MPEG-4 application in full search mode with three different test sequences **mobile** on the left, **foreman** in the middle and **news** on the right.

Figure 2 and 3 show also the impact in performance when using full search and optimized search algorithm. Comparing this factor shows a significant drop in cycles. One observation is the fact that the importance of using a good SAD is even more crucial in the case of full search, compared to the optimized search. As the number of times using SAD decreases when using the optimized search algorithm. The total improvement in execution time is not that significant but taken into account the ARM1136EJ has a width of only 32 bits, than this is a reasonable SIMD capability for embedded processors.

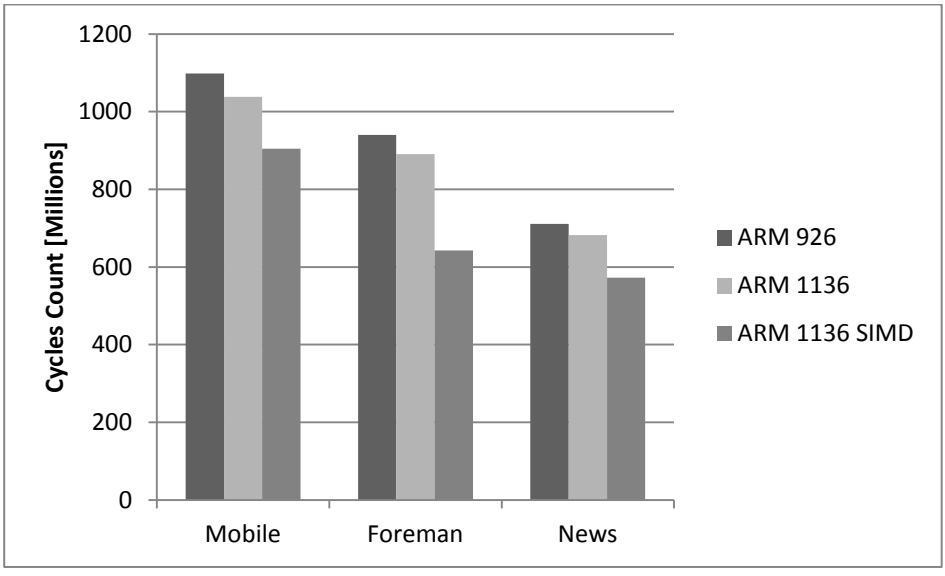


Figure 3. Cycle count for running MPEG-4 application in optimized search mode with three different test sequences **mobile** on the left, **foreman** in the middle and **news** on the right.

Figure 4 and 5 show the number of frames that can be executed in one second. This determines if the required performance of 30 Frames per second (F/s) is sufficient or not. The only time this can be achieved is when encoding the news sequence on the ARM1136EJ with SIMD extension. In the case of mobile the performance is around 20 F/s. When SIMD instructions are not used the execution time increases and none of the test sequences reaches the required 30 F/s. When compared with 12.5 F/s for news and around 8 F/s for mobile on ARM926EJ this shows a substantial improvement in the new of ARM1136EJ architectures. When looking at the difference between the full search and the optimized search, the most obvious observation is the impact of the search algorithm.

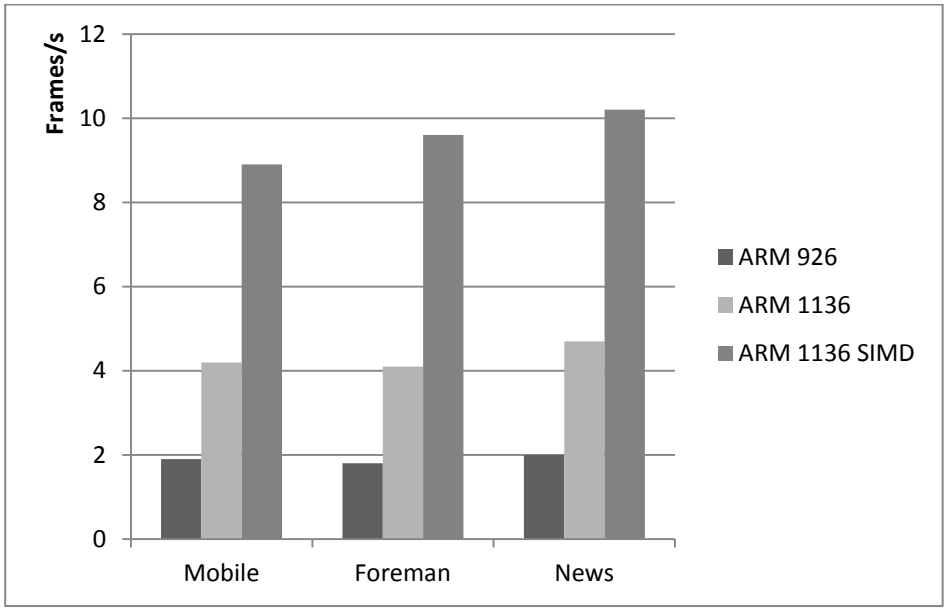


Figure 4. Frame rate for running MPEG-4 application in full search mode with three different test sequences **mobile** on the left, **foreman** in the middle and **news** on the right. It shows our proposed SIMD extension when run the same code as the one run on the ARM processors with SIMD SAD operation.

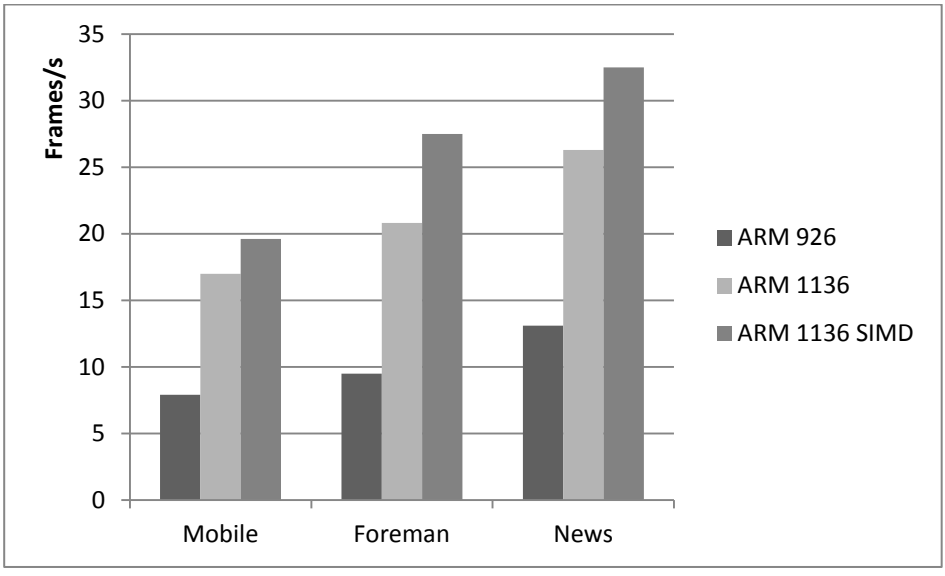


Figure 5. Frame rate for running MPEG-4 application in optimized mode with three different test sequences **mobile** on the left, **foreman** in the middle and **news** on the right.

In the case with full search there is no chance of achieving the target frame rate of 30 F/s. The best frame rate is 10 F/s when encoding news test sequence on the ARM1136EJ with SIMD extension. The impact of SIMD extension is even more noticeable when comparing with ARM1136EJ executions times with and without SIMD. The best frame rate ARM1136EJ can achieve is 4.3 F/s in full search and 25 F/s with optimized search.

Taking a look at the difference between ARM926EJ and ARM1136EJ we observe the increased impact of operating frequencies. This shows that need to run the processor at higher frequencies is necessary if one wants to encode streaming data. The important issue that rises is whether the increase in power consumption and area is justified. The power consumption is 0.48 W for ARM1136EJ compared to 0.15 W for ARM926EJ in the same 0.13 μm process including separate data and instruction caches.

When taking into account that the ARM1136EJ is running at 600 MHz and ARM926EJ is running at 300 MHz the important issue is the energy consumption. By increasing the power consumption but at the same time decreasing application's execution time even more, we save in total energy. This is will in fact be a better trade-off for handheld devices, as they are battery driven. We showed this in our previous work on SIMD extended architecture [10]. Regarding area the ARM1136EJ takes approximately 5.55 mm^2 compared to 5.0 mm^2 for ARM926EJ including instruction

and data caches [15].

We performed simulations on our own architecture running the same streaming video application and the same test sequences: mobile, foreman and news. The configuration of the encoder was the same as the ARM configuration. We performed the simulation both with zero-wait memory model and with specific memory configuration. This configuration assumes cache latencies for level1 caches 1 cycle, level2 cache 6 cycles and 45 cycles for memory. The results show with a more realistic memory model the cycle count increases by ~6%.

2.6 Discussion of the Results

The ARMv6 architecture shows potential to provide better performance for embedded processors. The architecture is evolving towards system-on-chip (SoC) design. The differences between the ARM926EJ and ARM1136EJ reflect the fact that the development of embedded processors is taking, even though cautiously, the same direction as high performance processors once took. They increase the number of pipeline stages to increase the clock frequency of the design. One important aspect to keep in mind is that the power and energy consumption for embedded device is going to be the main limiting factor. One has to keep the total energy consumption down, otherwise the cost of increasing the performance cannot be justified.

Traditionally the solutions chosen in the embedded domain have been ASICs with a set of different hardwired solutions. The main issue behind this decision has been predictability i.e., the full knowledge of the system functionality that need to be implemented. This fact made embedded system suitable for hardwired solutions. But as handheld devices have evolved to becoming a more open system, some of the predictability of the closed embedded systems has been lost. Thus the need for more flexible solutions has reason. At the same time the issue of energy is still dominant for handheld devices making it impossible to use just any solution. As embedded processors still have not got the performance required for running demanding applications such as video encoding, the need to boost the performance in an adequate way is important.

The introduction of a SIMD unit to embedded processors is a good way to both increase the performance and, at the same time, keep the power consumption down, as ARM shows [9]. We also see this in our own proposal [4], especially when one takes into account the total energy consumption, which is the main factor for handheld devices.

The ISA extension in processor introduces two favorable factors for platform design, flexibility and performance increase. This also reduces the complexity of designing complex heterogeneous ASICs. Thus reducing the design time and time to market for a platform. This trend in replacing as much as possible hardwired solutions as well as large DSP cores is quite significant. The replacement of hardware solutions has more to

do with flexibility as well as the issue of manufacturing costs of ASICs [18].

Replacing DSP core on the other hand has more to do with energy and area costs. Also the issue of intellectual property (IP) cost is a factor, where processor with enough SIMD performance can replace the DSP altogether [1, 11].

Now considering all this, one realizes that ARMv6 cannot provide enough performance to meet the needs of heavy applications such as video encoding. Therefore we see the need for an even more aggressive SIMD design. The only time the ARM1136EJ processor was able to achieve 30 F/s was when encoding the least computationally heavy test sequence news when using SIMD instructions. The introduction of a SIMD unit in ARM1136EJ is an important step in the right direction but it shows clear lack in performance when it comes to really heavy media applications such as video encoding. The need for more aggressive solutions with massive parallel architecture is obvious. When introducing this kind of parallelism, new issues such as memory structure and bandwidth become important. In our first study [10], we concluded that this is one of the main limiting performance factors.

2.7 Conclusion

In this paper we analyze and evaluate two generations of embedded processors in the same family to see the directions they have taken in terms of boosting the performance. We have chosen streaming media application as a benchmark for our investigation. The application provided to us is a full encoder with realistic behavior, making it suitable for benchmarking. We observe the direction embedded processors have taken is the same as high performance architectures took. Adding specific extension to the processor compensates for some existing shortcoming in performance.

The approach ARM has taken is to extend the ISA with SIMD instructions for specific operations such as SAD. This is a good way of increasing performance and still keeping the energy and area cost low. The focus is to efficiently handle highly parallel operations. This is a common feature of many media application. In our analysis we observed that the added SIMD extension in ARM1136EJ boosts the performance by 3 times, but the target performance is only achieved with the least demanding test sequence, news, with optimistic assumption of zero-wait high level memory.

In order to provide enough performance for MPEG-4 encoding we see a need to have a more aggressive SIMD architecture with wider data paths. This introduces some new issues among others a memory bandwidth bottleneck.

References

[1] R. Wilson, “Embedded CPUs take on media task”, EETimes, Okt. 2004.

[2] A. Gentile, D. S. Wills, “Portable Video Supercomputing”, IEEE Trans. Computers, vol. 53, no. 8, Aug.2004.

[3] D. Guevorkian, A. Launiainen, P. Liuha and V. Lappalainen, “Architecture for the Sum of Absolute Differences Operation”, in Proc. 2002 IEEE Signal Processing Systems, SIPS '02, pp. 57 –62.

[4] Intel, “MMX technology architecture overview”, Intel Technology Journal, Sept. 1997.

[5] BDTi, “Selecting Application Processors for Mobile Multimedia”, Berkeley Design Technology Inc., Jun. 2004 http://www.BDTI.com.

[6] P. Hus, K. J. R. Liu, “Software optimization of H.263 video encoder on Pentium processors with MMX technology”, in Proc. IEEE Multimedia and Expo, New York City, NY Aug. 2000.

[7] D. Etiemble, “Optimizing DSP and media benchmarks for Pentium 4: hardware and software issues”, IEEE 2002.

[8] http://www.mpeg.org/MPEG/MSSG/#source.

[9] V. Lappalainen, T. D. Hämäläinen and Petri Liuha, “Overview of Research Efforts on Media ISA Extensions and Their Usage in Video Coding” IEEE Trans. Circuit and System for Video tech., vol. 12, no. 8, Aug. 2002.

[10] A.R.Iranpour, K. Kuchinski, “Evaluation of SIMD Architecture Enhancement In Embedded Processors for MPEG-4”, in Proc. 30th EUROMICRO Digital System Design Conf. 2004.

[11] M. Baron, “MIPS Takes Aim at lo-cost DSP”, Microprocessor Report, Nov. 2004.

[12] D. Brash, The ARM Architecture Version 6 (ARMv6), ARM Limited, Cambridge, UK, Jan 2002, http://www.arm.com.

[13] ARM926EJ, ARM1136EJ technical manual, http://www.arm.com.

[14] D. Cormie, “The ARM11 Microarchitecture”, ARM Limited, Cambridge, UK, Apr 2002, http://www.arm.com.

[15] Cary D. Snyder, “ARM Family Expands At EPF”, Microprocessor Report, June 2002.

[16] http://www.standard.pictel.com/ftp/video-site/sequences/.

[17] Realview development suit version 2.1, http://www.arm.com.

[18] ITRC, International Technology Roadmap for Semiconductors, http://public.itrs.net.

Paper III

Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors[3]

Abstract. *In this paper we investigate the impact of different memory configurations on performance and energy consumption of the video encoding applications, MPEG-4 and H.264. The memory architecture is integrated with SIMD extended embedded processor, proposed in our previous work. We explore both dedicated memories and multilevel cache architectures and perform exhaustive simulations. The simulations have been conducted using highly optimized proprietary video encoding code for mobile handheld devices. Our simulation results show that the performance improvement of dedicated memories on video encoding applications is not very significant. The multilevel cache-based architecture processes approximately 17 frames/s compared to 19-22 frames/s for 512 KB dedicated on-chip zero-wait state memory. Thus it is difficult to justify using dedicated memory for this kind of embedded systems, when energy consumption and cost of implementation are also considered.*

[3] This paper is a reformatted version of *Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors*, in Proc. Embedded Computer Systems: Architectures, MOdeling, and Simulation (SAMOS VI), Samos, Greece, July 17-20, 2006.

3.1 Introduction

The video encoding applications implementing standards such as MPEG-4 and H.264 are computationally and memory intensive. These applications are becoming a dominant portion of today's computing workloads for handheld embedded devices. Since, embedded devices have limited energy supply and size they need to be designed carefully to fulfill these confined demands. With this in mind, these devices have other design challenges than high-performance designs. The key is to increase the performance just enough to meet the requirements with as little cost overhead as possible.

An important characteristic of video processing applications is the presence of data localities. This provides the possibility to use a special memory architecture that reuses data efficiently. Choosing the right memory solution is important in order to provide sufficient performance and manageable energy consumption. The memory solutions range from dedicated memories [1-4] to standard memory hierarchies [5-7].

The research on data reuse in media applications provides contradicting conclusions. Some authors argue that data reuse is ineffective for these applications [8]. They however concentrate on computational kernels only. Other authors draw different conclusions when the whole video processing application is considered [7]. Their paper confirms the existence of data locality for video encoding applications such as MEPG-4 for non-SIMD architecture, and the authors state that specific memory system optimizations fails to improve MPEG-4 performance. In this paper we examine Single Instruction Multiple Data (SIMD) enhanced embedded processor and video encoding applications specifically developed for mobile applications.

We use two proprietary video encoding applications provided by Ericsson AB specifically developed with focus on very low complexity algorithms, which influences both computations and memory traffic. This is important when evaluating memory architecture.

In our previous work [12] we have proposed SIMD extension for embedded processor to address the problem of high computation requirements for video encoding. In this paper we examine the impact of different memory architectures on performance and energy consumption of our architecture. We evaluate standard *multilevel cache hierarchy* against *dedicated memory* because the standard code can be run without rewriting that provides system flexibility.

The structure of the paper is as follows. Section 2 describes our video encoding applications. In Section 3 basic information of our extended processor architecture is given and in section 4 the memory architecture is discussed. Section 5 presents the method used in our approach. In section 6 we present and discuss our experimental results. In Section 7, we discuss related work and in Section 8 we give some concluding remarks.

3.2 Video Application

The two video standards used in our research are MPEG-4 [9] and H.264 [13]. Both are block based and one could view H.264 as the next step after MPEG-4. Looking at the optimized implementations of H.264/AVC encoders, time complexity is about 3.4 times higher for H.264 than MPEG-4 [14]. There are different phases involved in video encoding, such as Motion Estimation (ME), Motion Compensation (MC), Discrete Cosine Transform (DCT), quantization (Q) and variable length coding (VLC) [9]. The MPEG-4 and H.264 implementations selected for our research are full proprietary video encoding applications provided to us by Ericsson AB [11].

By profiling both MPEG-4 and H.264 video encoding applications we have identified the main computationally intensive operations. Our MPEG-4 application allows for full search (FS) and optimized search (OS) modes, with the last being more realistic for mobile devices. In MPEG-4 the most time consuming operations are in motion estimations Sum-of-Absolute-Differences (SAD), DCT as well as SAD_Intra. These operations account for 25-80% of the entire encoding time in case of MPEG-4 [15].

Two different H.264 implementations were evaluated. H.264 Ultra light (UL) comparable in quality to MPEG-4 and H.264 FAST comparable to the reference implementation [10]. The profiling of H.264 FAST and UL while performing encoding of the foreman test sequence shows the suitable operations for data parallelism account for approximately 40% of encoding time. These operations are Sum-of-Absolute-Transformed-Differences (SATD) and interpolation where they are significant part of the overall encoding time.

The main difference between our two implementations of H.264 encoder is the time complexity of the encoders and the quality of the encoded video sequence. The H.264 UL implementation is the simpler of the two and on average performs more efficiently than MPEG-4. The H.264 FAST encoder is more computationally demanding. This encoder performs well against the H.264 reference code [10] even though the time complexity of our encoder is significantly lower, approximately a speedup with a factor of 100 with an average bit-rate increase of less than 20%, than the reference encoder. This corresponds to approximately 700 times fewer SAD calls and 35 times fewer SATD calls [11].

The most often executed operation of video encoding, as identified by our profiling, use pixel arrays that represent frames. Macroblock (MB) is the main block of data where in MPEG-4 it consists of 8x8 pixels or 16x16 pixels. H.264 uses variable block sizes where macroblocks are partitioned into smaller MB 16x8, 8x8, 8x4 and 4x4. The frames are allocated in consecutive memory locations represented as pixel arrays. The allocated size for each frame is the frame size plus a border of 16 pixels surrounding the frame to deal with edge macroblocks when these are moved. The allocated memory for each section with screen size, QCIF (176x144) is 35 KB, CIF (352x288) is 122 KB

and VGA (640x480) is 343 KB.

3.3 Processor Architecture

For the purpose of this study we have extended an embedded processor (MIPS based) with a specialized SIMD unit. This unit is designed in such a way that it supports specific operations found in video encoding algorithms such as MPEG-4 and H.264. Media applications and in particular video encoding is well tailored for SIMD based solutions, as there is abundance of data-level-parallelism [8,16,17]. In addition, SIMD design is highly efficient in exploiting the structure and resources of the processor.

Our SIMD unit proposed in our previous work [12] is a pipelined unit with specific instructions that increase the overall performance. The baseline architecture contains a MIPS CPU with SIMD unit as well as the cache hierarchy and the main memory. The second architecture extends the baseline architecture with dedicated memory SIMDMeM. A more detailed schematic of the MIPS core and the SIMD unit microarchitecture is depicted in fig. 1. SIMD unit is integrated in the flow of the instruction pipeline of the processor. At the instruction decode stage, SIMD instructions are identified and redirected to SIMD unit. The SIMD unit executes load and arithmetic instructions. It has two vector registers VR1 and VR2 that can be configured either as 16X8-bits or 8X16-bits registers. The bandwidth of the memory interface is 64-bits, thus resulting in two or three load accesses for loading each vector register. Three accesses are needed for alignment of data.

Five arithmetic and five memory instructions have been added to the MIPS ISA to support the SIMD extension. The new instructions follow the same ISA as the other processor instructions. The load instructions perform the loading of vector registers (VR1 and VR2). The arithmetic instructions work on the two vector registers. Instruction SIMDSAD16 performs first 16 absolute value operations in parallel and then a tree of adders (together 15) sums all these values. SIMDSAD8 performs two 8x8 MB SAD operations. SIMDSATD first performs a Hadamard transform and then calculates SAD on the difference array. SIMDFIR performs a FIR filtering in half pixel interpolation. SIMDAVG performs average value for two pixel values for quarter-pixel calculation. The vector-processing unit is pipelined and has several stages, depending on used adders and technology. The speed-up for our SIMD using different SIMD instructions is approximately 6-7 times for SIMDSAD/SIMDSATD, three times for SIMDFIR and two times for SIMDAVG.

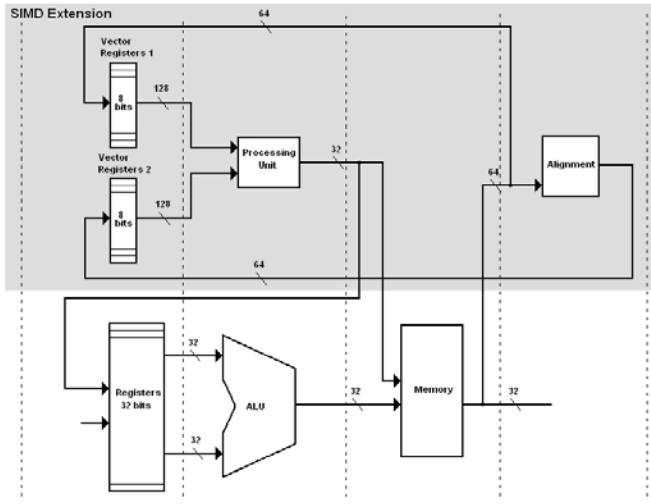


Figure 1. The proposed SIMD architecture.

3.4 Memory Architecture

The memory system and cache utilization stands out as one of the main issues, when introducing our SIMD support. As this affects many parts of the system, we need to investigate the architectural design tradeoffs. We investigated three different solutions, one using the standard memory hierarchy, the other introducing a zero-wait state separate memory for the SIMD unit, SIMDMeM, and a third using a dedicated zero-wait state frame memory. As the impact of this memory on the overall encoding performance was the focus, we evaluated a memory sufficiently large to hold all data we need.

We evaluated both the impact of level-1 and level-2 caches on the overall performance. Caches provide good performance for video encoding applications, since these applications have good spatial locality. Many procedures in these applications access data sequentially in blocks of 16 bytes.

Our SIMD memory SIMDMeM, acts as a tightly coupled memory (TCM), holding all data used by the SIMD unit. This provides a zero-wait state memory for SIMD calculations, thus removing memory latencies from the memory hierarchy. SIMDMeM also use the same address space as the main memory. We do not discuss any specific organization of this dedicated memory but our assumptions provide an ideal model. A real dedicated memory cannot provide better performance than the model used in our studies. As we will show even with this assumption, the overall performance of the encoder is not improved very much comparing to the standard cache hierarchy.

An alternative solution would be an on-chip zero-wait state frame memory. The

minimum size for this on-chip memory is dependent mainly on frame size and number of reference frames used. The memory footprint for our H.264 encoder with screen size of QCIF (176x144) and four reference frames is at least 512 KB. This solution reduces the energy costs of off-chip communication, at the same time, a 512 KB for on-chip fast memory might be difficult to justify in an embedded system.

3.5 Methodology

For verification of our architecture we used the two proprietary video encoding applications presented in section 2. Instruction Set Simulator (ISS), which is based on SimpleScalar toolset was used for the evaluation. This toolset provides an infrastructure for architectural modeling [18]. To estimate power we integrated the power estimation tool Wattch [19] into our system. The switching information for registers, functional units and buses was collected and used by Wattch for power calculation. The SimpleScalar cycle accurate model sim-outorder, modeling an in-order processor, with MIPS ISA has been chosen. We compile our video applications with the MIPS gcc compiler included in SimpleScalar toolset at optimization level –O3. Three memory configurations were used in our experiments: a separate level-1 instruction and data cache together with a unified level-2 cache, dedicated memory SIMDMeM, and on-chip frame memory. Table 1 illustrates configuration of the system with the underlined values representing the memory architecture configuration proposed after our investigation presented in section 6.

We evaluated two different architectures, baseline SIMD extended with standard memory hierarchy and SIMD extended with dedicated memory. The processor clock speed was set at 650 MHz in all the simulations with 90 nm process power model. The energy model for the off-chip memory includes the memory and communication energy consumption. The memory hierarchy latency for the system is 6 cycles for level-1 cache miss for the first chunk of data and 1 cycle for the consecutive chunks. A cache level-2 miss gives 45 cycles latency for the first chunk of data and 5 cycles for the consecutive data chunks when fetching data from the main off-chip memory.

Table 2 illustrates the configuration chosen for the MPEG-4 and H.264 encoders. The screen resolutions chosen were QCIF (176x144) and CIF (352x288). H.264 is restricted with the screen resolution of QCIF as our encoder for the moment supports this size. Test sequences chosen in our experiments were foreman, mobile and news [20]. The main difference between these sequences is the amount of processing they need to encode the sequences. The mobile sequence is the most demanding sequence in terms of processing. In order to measure the overall performance of the system we used frames per second, which in our case is more relevant as we are performing video encoding. As we are dealing with handheld, battery driven embedded devises we use total energy consumption rather than power consumption. For evaluating cache

performance we use miss rate, which is a common practice. But as we will point out later, blindly using miss rate alone can be misleading, as cache accesses influence the total energy consumption.

Table 1. Cache architecture, dedicated memory SIMDMeM and on-chip frame memory with the chosen size and configurations (underlined in the table).

	Size (KB)	Line size (Bytes)	Associativity	Replacement policy
Inst. Cache	8/**16**/32/64	16/**32**	**2**	**LRU**
Data Cache	8/16/**32**/64/128/256/512/1024	8/16/**32**	1/2/**4**/8/16	**LRU**
Unified Cache	64/**128**/265/512/1024/2048	32/**64**/128	**4**	**LRU**
SIMDMeM	**128**	-	-	-
On-chip Frame memory	**512-768**	-	-	-

Table 2. MPEG-4 and H.264 configuration.

	Screen size	Quantization	Search algorithm	Comments
MPEG-4	QCIF/CIF	15	Full Search/ IGRADD	Half-pixel enabled
H.264 Ultra Light (UL)	QCIF	30	IGRADD	Ref. frames 4
H.264 FAST	QCIF	30	IGRADD	Ref. frames 4

3.6 Experimental Results and Discussion

In sub-section 6.1, we evaluate the level-1 instruction and data cache size and their configurations for video encoding. In sub-section 6.2 we present our evaluation of the level-2 cache and its impact on performance and cache miss rate. Sub-section 6.3 deals with the energy consumption of the architecture. The results obtained in section 6.1-6.3 are then used to select an appropriate cache configuration when comparing with dedicated memory. In sub-section 6.4, we present the performance results for encoding applications on the evaluated architectures for both standards cache hierarchy and dedicated memory. Finally, we discuss experimental results and their implications in sub-section 6.5.

3.6.1 L1 cache configuration

To find the optimal cache configuration for our two encoding applications we performed extensive simulations for different cache configurations. We have chosen

separate instruction and data cache architecture. The evaluated data cache sizes for level-1 cache were 8, 16, 32 and 64 KB, which are the most common sizes used. The increased data cache size has positive effect on encoded frames per second (frames/s) as shown in fig. 2a, but this comes at the expense of increased energy (fig. 5a) as discussed later. Based on the analyses of miss rate for level-1 data cache, fig. 2b, we can conclude that the level-1 data cache already at 32 KB has a miss rate between 1.2-3.1% for all applications. This provides a performance of 30 frames/s for most applications except H.264 FAST.

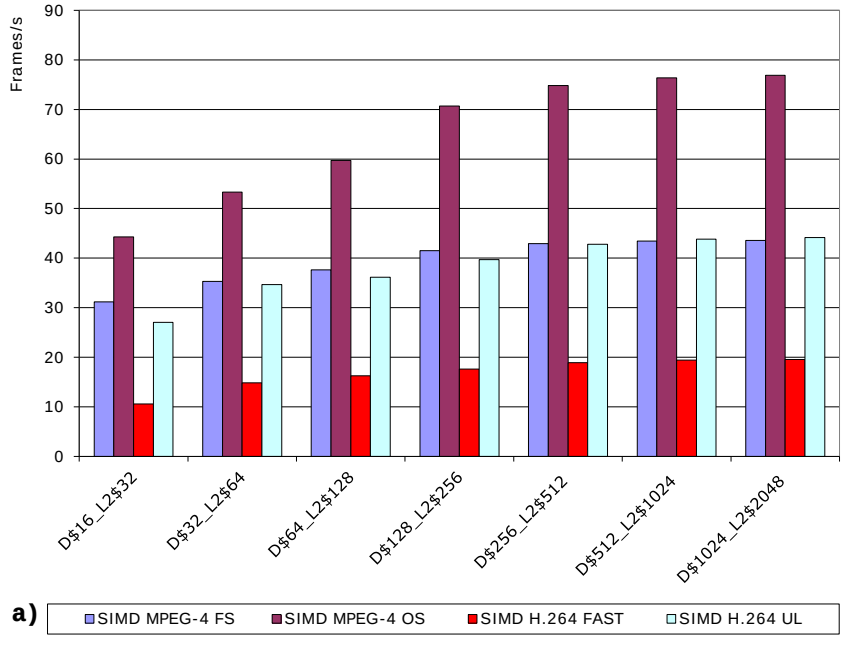
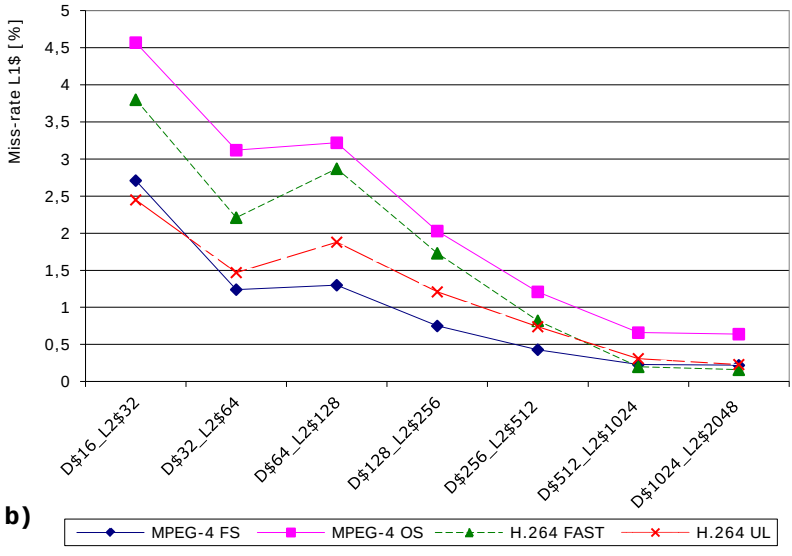


Figure 2. a) Frame rate for MEPG-4 and H.264 with different level-1 cache sizes.



b)

Figure 2. b) Miss rate for MEPG-4 and H.264 with different level-1 data cache sizes.

Fig. 3a shows performance for 4, 8 and 16 KB instruction cache sizes. Caches larger than 16 KB are not shown in figures, but we have observed that there is no significant miss rate improvement and are not realistic for embedded systems. Looking at the frame rate depicted in fig. 3a, going from 8 to 16 KB instruction cache gives a significant improvement for our application. An important factor is the level-1 instructions cache miss rate, which is as high as 29% for 4 KB and 20% for 8 KB going down to 4.3% for 16 KB instruction cache.

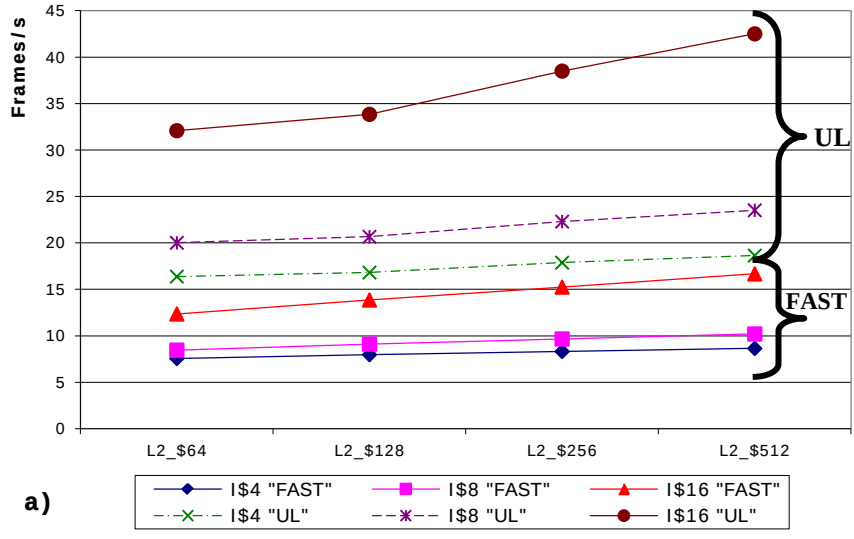


Figure 3. a) Frame rate for H.264 FAST and UL with different level-2 and level-1 instruction cache sizes.

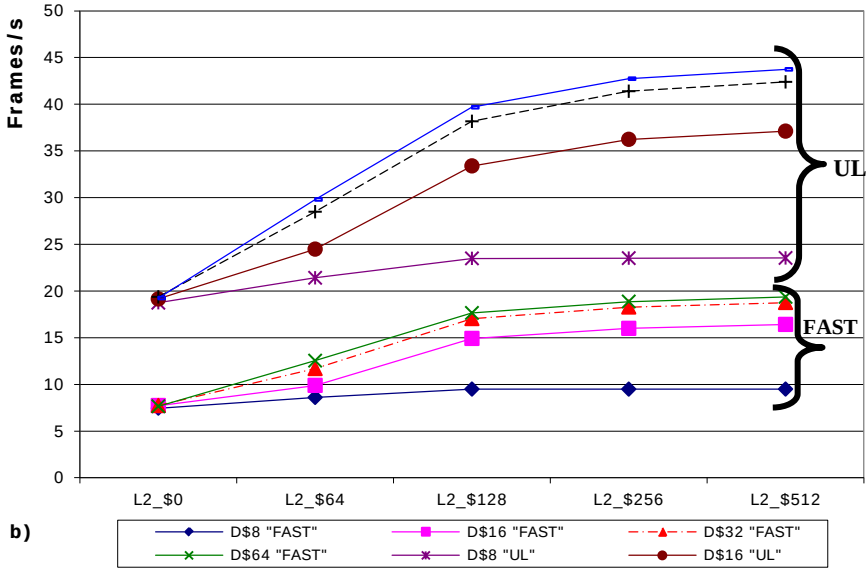


Figure 3. b) Frame rate for H.264 FAST and UL with different level-2 and level-1 data cache sizes.

Cache associativity is another key issue for cache performance. The number of cache accesses decrease when we go from direct mapped to 2-way, 4-way associativity. Our experiment shows that going beyond this to 8-way and above provides no significant improvement. As our results show, the low miss rates in level-1 cache indicates the high reuse of data in level-1 cache. The main bandwidth bottleneck is between level-1 cache and processing unit. In our architecture a 64-bits bus handles this. Our simulations indicate 16 KB being right size for level-1 instruction and 32 KB for level-1 data cache.

3.6.2 L2 cache and its impact

Fig. 3b illustrates the impact of level-2 cache for encoding foreman test sequence. This test sequence can be considered as good average since similar results were observed for both mobile and news test sequence. As in previous sub-section the presented results are for H.264 encoding. The results of MPEG-4 indicate a similar pattern. We observe the potential benefits of reducing the size of level-1 cache with small performance degradation on the overall encoding. If the size of level-1 is below 8 KB the size of level-2 cache has no impact on overall performance. Cache level-1 of 16 KB and above provides significant improvement with added level-2 cache. The optimal size, when taking into account miss rate as well as energy consumption and performance, of level-2 cache is 128 KB. This is true for all test sequences.

The impact of introducing a level-2 cache, which is significantly slower but larger than level-1 cache, is apparent on overall performance. We observe a miss rate improvement when going from 20-16% for 64 KB level-2 cache to 8-4% for 128 KB and below 1.5% for 512 KB. An important issue is the impact of level-1 cache on level-2 cache. The observation made for instruction cache, that a larger level-1 cache gives a higher miss rate in level-2 cache, is also true for level-1 data cache.

In fig. 3b there is a break at 128 KB where the curve flattens and we observe less noticeable improvement with increased level-2 cache size. The same results were also obtained and verified for encoding MPEG-4, but due to space limitations we only present H.264 encoding results.

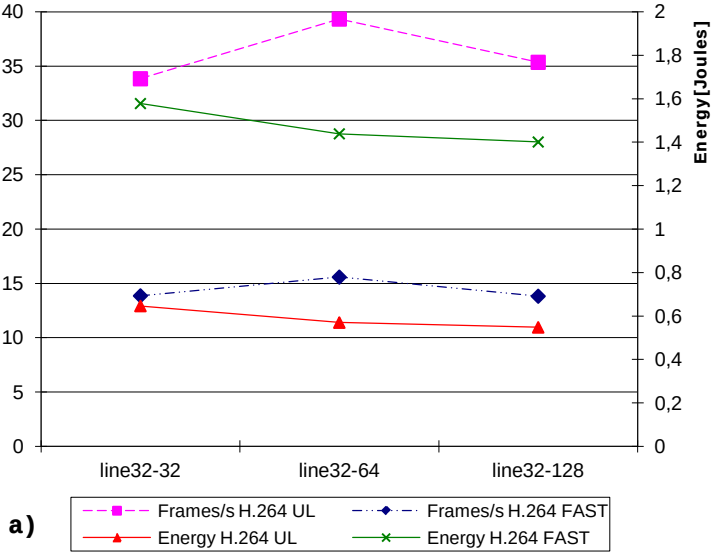


Figure 4. a) Frame rate and energy consumption for different level-2 cache line sizes.

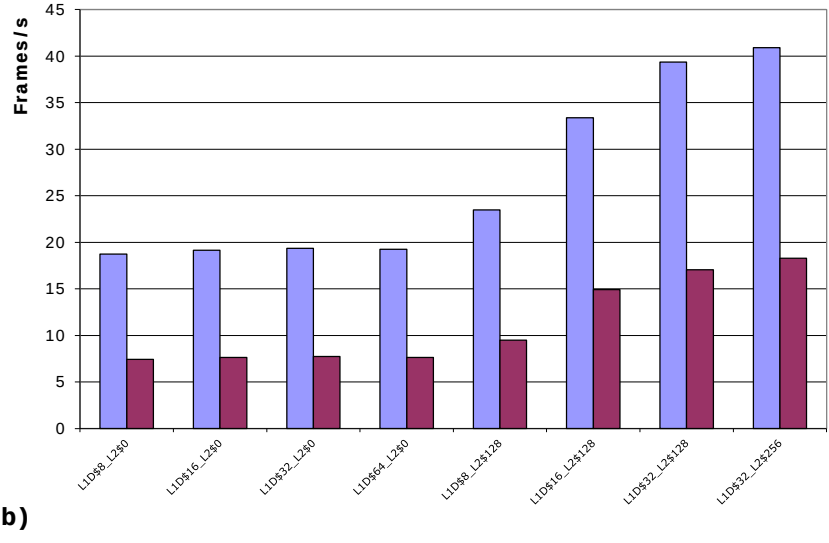


Figure 4. b) The impact of level-2 cache on performance while encoding foreman with H.264 UL and FAST.

As shown in fig. 4a the optimal line size for level-2 cache, which in our study was 64-bytes. The positive impact of increased level-2 line size both saves energy as well as lowers the miss rate. At the same time the number of accesses is almost identical. This has more impact on the overall system performance than level-1 line size. Going beyond 64-bytes does not give any significant improvement on the overall performance and has negative impact on the overall energy consumption.

Fig. 4b shows the overall improvement in video encoding performance that can be obtained by introducing a level-2 cache. The significant performance jump can be observed when we use level-2 cache together with a large enough level-1 cache. In our case this is at 32KB for level-1 data cache and 128 KB level-2 unified cache. Going beyond this has no significant overall improvement.

3.6.3 Energy consumption

Fig. 5a depicts the total energy consumption of the system with caches and off-chip memory. The total energy consumption includes also our SIMD unit but it is usually lower for SIMD enhanced architecture even though we have introduced a new component in the processor architecture [11]. The energy consumption of different cache configurations while performing video encoding on the foreman test sequence shows that the optimal point is at 128 KB level-2 cache, 32 KB level-1 data cache and 16 KB level-1 instruction cache sizes. As can be seen this is true both for H.264 FAST and H.264 UL, similar results were obtained for MPEG-4 as well. Fig. 5b shows the energy consumption when encoding MPEG-4 as well as H.264 with the final memory architecture. The overall energy consumption is almost identical for both SIMD and SIMDMeM.

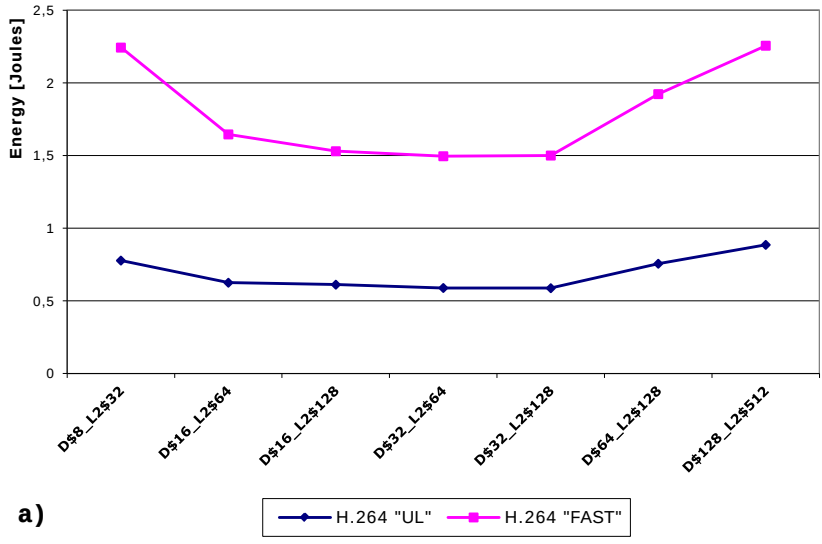


Figure 5. a) Energy consumption for H.264 (FAST) and (UL) for encoding foreman test sequence with different level-2 data cache sizes.

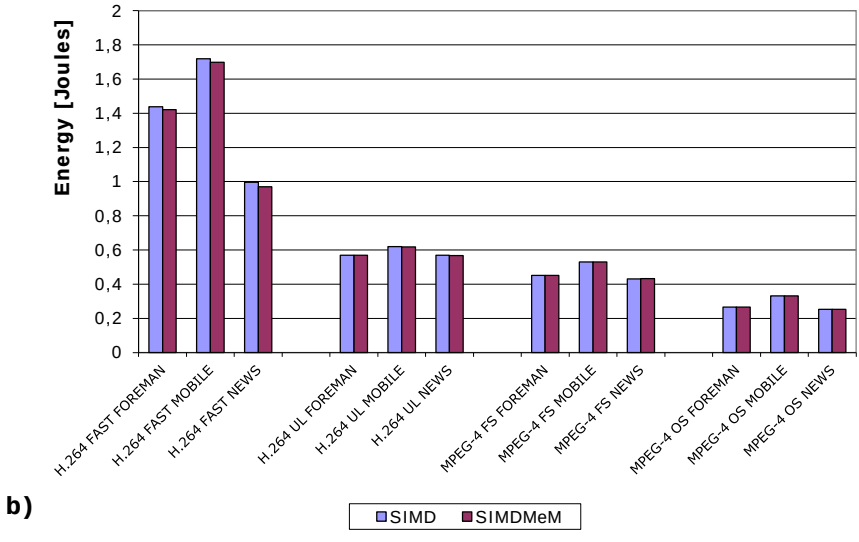


Figure 5. b) Energy consumption for MPEG-4 (FS) (OS) and H.264 (FAST) (UL) for test sequences foreman, mobile and news.

3.6.4 Dedicated memory vs. cache

The three memory configurations were standard cache hierarchy as discussed earlier and dedicated memory SIMDMeM for SIMD unit, as well as dedicated frame memory. The optimal cache configuration we found in previous sub-sections was used for our evaluation (see table 1). Fig. 6 shows the performance of our two memory architectures for H.264 FAST and UL as well as MPEG-4 when encoding the three different test sequences foreman, mobile and news.

The performance of SIMD and SIMDMeM are almost identical which shows the impact of adding a dedicated memory to SIMD unit has no significant impact over standard cache memory organization. With regards to energy consumption in fig. 5b we do not see any significant difference between the two memory architectures.

We have also evaluated the most optimistic data memory hierarchy, where all frame data used for encoding is in dedicated zero-wait state frame memory. The encoding of foreman results in 22 frames/s compared to 17.5 frames/s for SIMD and 18.6 frames/s for the SIMDMeM solution. The increased energy consumption from using a standard memory hierarchy with off-chip frame memory compared to using an on-chip zero-wait state frame memory is 0.35 J. This includes off-chip memory and communications for our chosen standard memory hierarchy configuration (see table 1). As stated before, the main arguments against an on-chip solution is the added costs in terms of size and practicality of having at least 512KB for on-chip zero-wait state memory. The justification for this solution in an embedded system is extremely hard especially when the gains are still relatively small.

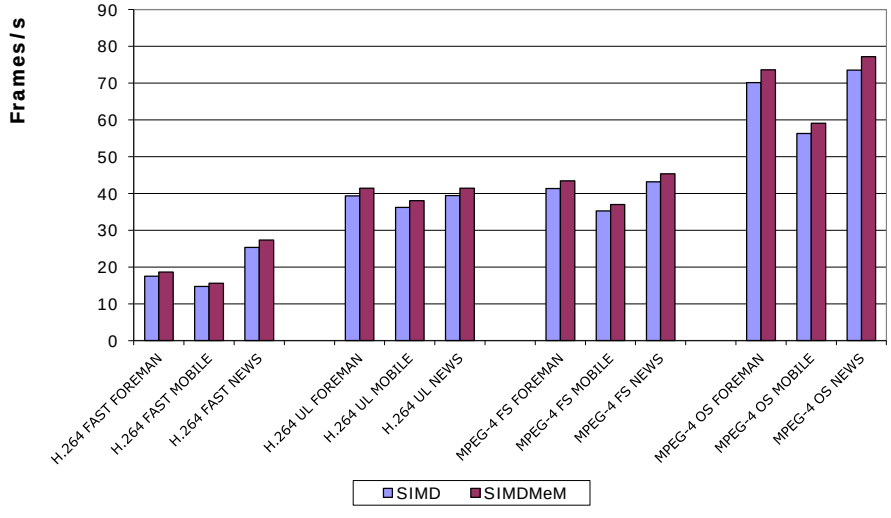


Figure 6. Frame rates for MPEG-4 (FS) (OS) and H.264 (FAST) (UL) while encoding the test sequences foreman, mobile and news.

3.6.5 Summary

The addition of dedicated memory for the SIMD unit has no significant benefit when performing video encoding, both for H.264 as well as MPEG-4, and regardless of the test sequence used. The benefit of increasing the size of level-1 data cache beyond 32 KB has no substantial improvements on the overall performance, as well as it may even reverse affect for the energy consumption. The introduction of level-2 cache, to hide the latencies between the level-1 cache and main memory, has more significant impact on the overall performance as well as energy consumption. This has also the positive side effect of being able to reduce level-1 cache size. In terms of using a dedicated SIMDMeM or dedicated on-chip frame memory the increase in performance is relatively small only by 4.5 frames for the dedicated frame memory.

3.7 Related Work

Most work on video encoding has been done using kernels [8,21], or using non-optimized code [3,4,7,22,23]. This approach makes it difficult to draw right conclusions for how an entire video encoding application behaves. In our study we use proprietary video encoding applications, ensuring that we have correct workloads for evaluation of our memory architectures.

The high bandwidth requirements of video encoding applications are important architectural design issues [22]. Multilevel caches, together with special instructions for computationally intensive application kernels, are discussed in [21] as important performance boosters. The authors of [6,7] propose to use bandwidth hierarchy to address the memory bandwidth problem. By removing the latency through usage of memory hierarchy the performance degradation was negligible and thus illustrated the potential of balanced memory architecture. We also use cache hierarchy but we test it with the SIMD unit that has higher bandwidth requirements. Our approach of using the standard memory hierarchy has the added benefit of not needing to optimize the data placement and having the added cost of dedicated memory.

Utilizing a level-2 cache has been a performance improvement factor in high performance processors. As more computationally demanding applications are executed on embedded systems, level-2 cache has been proposed for embedded domain as well. There are though not many studies done in this regard. In [6] the authors briefly discuss CPU utilization and transaction traffic when introducing level-2 cache for video decoding. Their finding is that both CPU utilization and transaction traffic decrease with increased level-2 cache size. In [7], the authors study performance of non-SIMD high performance processors for MPEG-4. Their architectures utilize large 1-8 MB level-2 cache, which improves the overall performance through reduction of traffic to main memory. In none of these works the emphasis has been on evaluating the actual impact of level-2 cache. The work has been on high performance general purpose processors using MPEG-4 reference code. In [6] there is a study on

performance improvement for embedded processors when introducing level-2 cache. This work looks at MPEG-4 decoding which has some similarities to encoding but is much less performance demanding. We use video encoding and an embedded processor with an SIMD unit. This puts different requirements on memory bandwidth.

Dedicated memories have been proposed to improve performance of application specific systems, for example in [3,4]. The authors in [24] propose a HiBRID multi-core system on chip architecture with 4KB dedicated memory to compute macroblocks. In our work we have evaluated dedicated memory architectures against multilevel cache hierarchies.

3.8 Conclusions

In this paper we have performed extensive simulations on our SIMD extended processor and show that using standard multilevel cache hierarchy achieve almost the same performance as a dedicated memory for the SIMD processing unit for video encoding. As video encoding is highly data centric the importance of a well-balanced memory is crucial. An important issue for this exploration is the use of realistic application workloads specifically implemented for handheld embedded devices when exploring different design trade-offs. We examine two solutions, one that utilizes the standard cache hierarchy (two levels) and the other one that uses a dedicated zero-wait-state memory. Our results show, against common belief, that the use of the standard cache based architecture achieves almost the same performance as SIMD dedicated memory architecture for full video encoding applications. We have made conservative assumptions in our energy models for dedicated memory but the overall difference in energy consumptions was negligible.

References

[1] V.A. Chouliaras et al., "A Multi-Standard Video Accelerator based on a Vector Architecture," IEEE Trans. Consum. Elec., Vol.51, No.1, Feb. 2005.

[2] J.L.Nunez. and V.A. Chouliaras, "High-performance Arithmetic Coding VLSI Macro for the H264 Video Compression Standard," IEEE Trans. Consum. Elec., Vol.51, No.1, Feb. 2005.

[3] Y.-W.Huang, B.-Y.Hsieh, T.-C.Chen and L.-G.Chen, "Hardware Design for H.264/AVC Intra Frame Coder," in Proc. of IEEE ISCAS'04, Vol. 2, II-269-272, 2004.

[4] R.G.Wang, J.T.Li and C.Huang, "Motion Compensation Memory Access Optimization Strategies for H.264/AVC Decoder," in Proc. of IEEE ICASSP'05, Vol. 5, pp.97-100, 2005.

[5] A.Stevens, "Level 2 Cache for High-performance ARM Core-based SoC System," White-paper ARM, Jan. 2004, http://www.arm.com/.

[6] A.Asaduzzaman et al., “Cache Optimization for Mobile Devices Running Multimedia Applications,” in Proc. IEEE ISMSE’04, pp. 499-506, 2004.

[7] S.A.McKee, Z.Fang and M.Valero, “An MPEG-4 Performance Study for non-SIMD, General Purpose Architectures,” in Proc. of IEEE ISPASS 2003, pp. 49-57, 2003.

[8] J.D.Owens et al., “Media Processing Applications on the Imagine Stream Processor”, in Proc. of IEEE ICCD’02, pp. 295-302, 2002.

[9] MPEG-4: ISO/IEC JTCI/SC29/WG11, “ISO/IEC 14469:2000-2: Information on technology-coding of audio-video objects–Part 2:Visual,” ISO/IEC, Genf, Switzerland, Dec. 2000.

[10] H.264/AVCSoftwareCoordination,JM,http://iphome.hhi.de/suehring/tml/

[11] C.Priddle, “H.264 video encoder optimization with focus on very low complexity algorithms,” M.S. thesis, Uppsala University, April 2005.

[12] A.R..Iranpour and K.Kuchcinski, “Evaluation of SIMD Architecture Enhancement in Embedded Processors for MPEG-4,” in Proc. IEEE DSD’04, Sep. 2004.

[13] Joint Video Team (JVT) of ISO/IEC MPEG, ITU-T VCEG “Text of ISO/IEC 14496 10:2004 Advance Video Coding Standard (second edition)”, ISO/IEC JTC1/SC29/WGII/N6359, Munich, Germany, March 2004.

[14] V.Lappalainen, et al., “Performance of H.26L Video Encoder on General-Purpose Processor,” Kluwer Journal of VLSI Sig. Proc., Vol. 34, No. 3, pp. 239-249, 2003.

[15] A.R.Iranpour and K.Kuchcinski, “Analyses of Embedded Processors for Streaming Media Applications,” in CAECW-8, Feb. 2005.

[16] V.Lappalainen, T.D.Hämäläinen and P.Liuha, “Overview of Research Efforts on Media ISA Extentions and Their Usage in Video Coding” IEEE Trans. Circuit and System for Video tech., Vol. 12, No. 8, Aug. 2002.

[17] S.Vassiliadis, B.Juurlink and E.Hakkennes, “Complex Streamed Instructions: Introduction and Initial Evaluation” in Proc. 26th Euromicro Conference, Vol.1, pp. 400-408, 2000.

[18] T.Austin et al., “SimpleScalar: An infrastructure for computer system modeling,” IEEE Computer, Vol. 35, Issue 2, pp. 59-67, Feb. 2002.

[19] D.Brooks, V.Tiwari and M.Martonosi, “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,” in Proc. ISCA’00, pp. 83-94, June 2000.

[20] Test sequences, http://www.chiariglione.org/mpeg/

[21] F.Franchetti, S.Kral, J.Lorenz and C.W.Uberhuber, “Efficient Utilization of SIMD Extensions,” in IEEE Proceedings, Vol. 93, No. 2, Feb. 2005.

[22] J.-C.Tuau, T.-S.Chang and C.-W.Jen, “On the Data Reuse and Memory Bandwidth Analysis for Full-Search Block-Matching VLSI Architecture,” in IEEE Trans. Circuit and Syst. For Video tech., Vol. 12, No.1, Jan. 2002.

[23] C.-Y.Cho, S.-Y.Huang and J.-S.Wang, “An Embedded Merging Scheme for H.264/AVC Motion Estimation,” in Proc. of ICIP 2003, Vol. 1, I-909, 2003.

[24] H.J.Stolberg et al., “HiBRID-SoC:A Multi-Core SoC Architecture for Multimedia Signal Processing” Journal VLSI Signal Processing System, Vol. 41, pp. 9-20, 2005.

Paper IV

Performance Improvement for H.264 Video Encoding using ILP Embedded Processor[4]

Abstract. *In this paper, we examine the impact of instruction level parallelism (ILP) on the full H.264 video encoding application and give quantitative performance measures of a superscalar architecture. Most research efforts have concentrated on the data intensive parts, such as kernels but these are taking less time from the entire execution as encoders are using new, more efficient algorithms. This important fact cannot be neglected since new video encoding standards have been proposed and the amount of other than data intensive computations has increased significantly. We observed significant improvement for the entire application when using superscalar architecture with out-of-order execution scheme. Tradeoffs in superscalar performance are also evaluated with combinations of measurements from SimpleScalar simulator.*

[4] This paper is a reformatted version of *Performance Improvement for H 264 Video Encoding using ILP Embedded Processor*, in Proc. of the 9th Euromicro Conference on Digital System Design, Cavtat/Dubrovnik, Croatia, August 30th - September 1st, 2006.

4.1 Introduction

The importance of handling different applications with high degree of computation intensity is a key issue for handheld embedded devises, such as mobile phones and PDAs. One such category of applications are video encoders. These applications put huge demands on the entire system, processor, memory, buses and other parts. Modern video encoding applications, such as H.264, used in handheld devices, utilize optimized algorithms that have very different execution profile than standard reference encoders.

There is a common belief, that in video encoding applications the data level parallelism (DLP) dominates and other parallelisms, such as instruction level parallelism (ILP), are of less importance. There are important issues of DLP in video encoding applications that need to be examined deeper. First, data level parallelism makes execution of parallel sections of the application faster. This makes the serial sections become more dominant. Second, we have through profiling, in our previous work, identified that the amount of DLP in optimized video encoders is decreasing. Therefore, only concentrating on encoding kernels, as most researchers do, does not provide enough performance improvements since the kernels are responsible for a shrinking part of the execution time. In this situation, we have to find ways to seek more parallelism in other parts of the encoding applications to improve the overall performance.

The instruction level parallelism, (ILP) which exist in video encoders, is not negligible, especially as the amount of control code in newer standards, such as H.264, is increasing because of more advance compression techniques. This fact together with the issues regarding DLP, makes ILP an obvious target for seeking increased amount of parallelism.

There are two approaches for breaking the single-instruction-per-cycle bottleneck, through usage of ILP, either by using superscalar processors or very long instruction word (VLIW) architectures. These two categories exploit ILP, statically or dynamically. For VLIW processors [1] ILP is found statically during compile time by the compiler. Dynamic ILP, on the other hand, exploits either in-order or out-of-order (OoO) hardware scheduler at runtime. Here we examine dynamic ILP, as we want to combine ILP together with DLP provided by our single instruction multiple data (SIMD) extended architecture proposed in our previous work [2, 3]. Intuitionally one can observe that increased DLP reduces amount of existing ILP. We also observed this but the amount of reduction was negligible.

In this paper we try to explore if there is enough parallelism to justify the use of out-of-order superscalar in combination with SIMD architectures in embedded domain. The goal is to encode 30 frames/s at CIF screen resolution with limited increase in size and energy.

The structure of the paper is as follows. Section 2, describes our video encoding applications. In section 3 basic information of our superscalar SIMD extended processor architecture is given and in section 4 the method used in our approach is presented. In section 5, we present and discuss our experimental results. In section 6, we discuss related work and in section 7, we give some concluding remarks.

4.2 Video Application

The main video standard used in our research is H.264. This standard is for wide variety of areas, such as videoconferencing, and it is also the main video standard recommended by 3GPP standardization group in release 6 [4]. The H.264 standard has many similarities with MPEG-4 as they are both block-based. One could view H.264 as the next step after MPEG-4. Looking at the optimized implementations of H.264 encoders, time complexity is about 3.4 times higher for H.264 than MPEG-4 [5].

H.264 was developed by Joint Video Team (JVT) and is a hybrid of the two existing video coding standards, the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Pictures Experts Group (MPEG) [6]. H.264 has dramatically reduced bit-rate while achieved 50% better compression, without compromising quality. The main added features are variable block sizes, where macroblocks (MB) can be partitioned into smaller blocks of size 16x8, 8x8, 8x4 and 4x4. Multiframe Prediction (MP) and quarter-pixel resolution are other features added to improve the quality of the final coded video sequence. The MP provides the usage of more than one previous frame as reference for motion estimation (ME). Another improvement is the use Hadamard transform in the SAD computation. In our architecture we use therefore SATD, a special instruction to compute it instead of SAD. In H.264 ME there is half-pixel interpolation, which is performed using a 6 tap FIR. The quarter-pixel is evaluated by averaging two half-pixel values.

The H.264 application we use is the proprietary software provided to us by Ericsson AB. The H.264 implementation evaluated, performs well against the reference implementation in terms of quality [7, 8], even though the time complexity of our encoder is significantly lower, approximately a speedup of 100 times with an average bit-rate increase of less than 20%, than the reference encoder. It uses optimized search mode and is configured for running screen size CIF (352X288) at 30 frames per second.

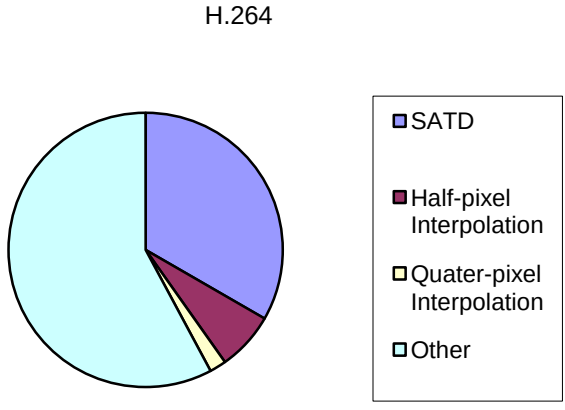


Figure 1. Profiling, the total execution time for H.264, whilst encoding test sequence foreman with 4 reference frames

Figure 1 presents the profiling of H.264 while performing encoding of foreman test sequence [9]. As can be seen, SATD and interpolation are the main operations where data parallelism exists. It significantly influences overall encoding time but other computations stand for nearly 60% of the total time. They cannot be made faster using specialized DLP solutions but ILP might provide a speedup for these parts. We have observed a significant difference between reference H.264 encoder and encoder written with focus on embedded handheld devices. This means that accurate and relevant architectural designs need realistic applications and crucial for media applications.

4.3 Processor Architecture

In order to achieve the target performance requirement of 30 frames/s we combined the SIMD architecture, proposed in our previous work [2], with superscalar architecture. The SIMD unit provides the DLP capabilities and the superscalar architecture introduces ILP to the system. The superscalar architecture could either be run as an in-order or out-of-order (OoO) processor. In the case of in-order the instructions are semi-dynamically scheduled, which means that instructions are continuously issued on non-dependent memory stalls. The OoO architecture uses a central instruction window for dynamic scheduling. The Design uses a Register Update Unit (RUU) and a Load/Store Queue (LSQ) to reschedule the instructions [10].

For the purpose of extracting DLP we have extended an embedded processor (MIPS based) with the specialized SIMD unit. This unit is designed in such a way that it supports specific operations found in video encoding algorithms for H.264. Our

profiling has identified specific parts of the application where there is large amount of data level parallelism. The SIMD unit proposed in our previous work [2] is a pipelined unit with specific instructions that increase the overall performance and provide instructions for H.264.

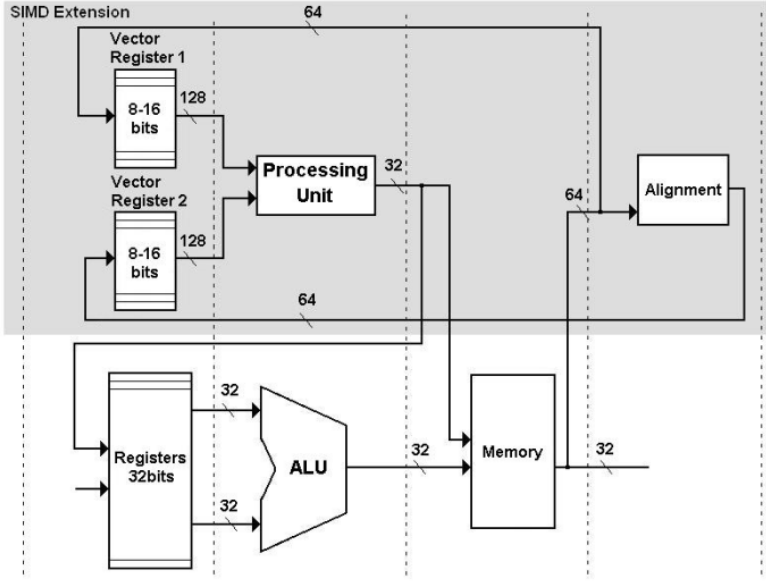


Figure 2. The proposed SIMD architecture. The connection from memory to the extension is 64-bit wide. The 64-bits coming from memory are in consequent addresses thus they can be read in one cycle.

A detailed schematic of the MIPS core and the SIMD unit microarchitecture is depicted in fig. 2. SIMD unit is integrated in the flow of the instruction pipeline of the processor. At the instruction decode stage, SIMD instructions are identified and redirected to SIMD unit. It executes load and arithmetic instructions. The SIMD unit has two vector registers VR1 and VR2. Each vector register can be configured either as 16 8-bits or 8 16-bits registers, depending on the SIMD arithmetic instruction. The bandwidth of the memory interface is 64-bits, thus resulting in two or three load accesses for loading each vector register. Three accesses are needed for alignment of data. Alignment is performed on the current frame macroblock, which is done by reading three 8-bytes from memory. These 24 bytes are then shifted left so that they become byte aligned, i.e., we get two 8-byte aligned data.

In this paper, we extended our previous SIMD architecture with the capability of 2D SAD computations. The instructions supporting these computations perform a SAD

operation on the entire macroblock (MB), thus reducing even more control code overheads.

The SIMD instructions follow the same ISA as the other processor instructions. SIMDLD16, SIMDLD8, SIMDLD16T, SIMDLDFIR and SIMDLDAVG are load instructions, which perform the loading of vector registers (VR1 and VR2). SIMDSAD16, SIMDSAD8, SIMDSATD, SIMDFIR, SIMDAVG are the arithmetic instructions performing the computation by taking VR1 and VR2 as input operands and writing the results back in the target register. The new 2D SAD instruction SIMDSAD2D work in the same way as previous instructions while, in addition, it preloads the next macroblock lines. The instruction has also the stride for fetching the next MB line and calculates the SAD result for the whole MB and thus resulting in less memory accesses.

For increasing Instruction level parallelism (ILP) we extend the processor width by 2, 4, 8 and 16 ways. In order to achieve a balanced superscalar design we have the same size fetch, decode, issue and commit width. The processor uses central instruction window for OoO processing, which requires less storage than reservation station. This central window can be implemented in different ways such as, dispatch stack, register update unit or reorder buffer. Our processor has been implemented using a Register Update Unit (RUU) and a Load/Store queue (LSQ). The size of the RUU and LSQ is important concerning the performance of the OoO unit. The RUU is a simpler implementation of central window that avoids the complexity of compressing the instruction window, as is the case with dispatch stack. The RUU operates as FIFO buffer with decoded instructions being placed on top of the FIFO and results being written to register file at the bottom of the FIFO. An important characteristic of RUU, compared to dispatch stack, is that an entry is not removed when the instruction is issued, but rather it is removed when it reaches the bottom of the FIFO. This result in a simpler design but the drawback is that it makes it sensitive to small instruction window sizes. This could also indirectly result in stalls in the decoder, if an instruction entry reaches the bottom of the RUU and the instruction has not been issued.

4.4 Methodology

The focus of this research was to evaluate the impact of different architectural techniques for extracting instruction level parallelism (ILP) on the full video encoding application and also to combine it with our proposed SIMD extended architecture for data level parallelism. To verify our architecture we used a proprietary video encoding application presented in section II. Instruction Set Simulator (ISS), which is based on SimpleScalar toolset was used for our evaluation. This toolset provides an infrastructure for architectural modeling [11]. To evaluate power we integrated the power estimation tool Wattch [12] into our system. The power model used for the architecture was also executed for the SIMD instructions. The switching information

for registers, functional units and buses was collected and used by Wattch for power calculation. The cycle accurate model sim-outorder with MIPS ISA has been chosen.

We compiled our video applications with the MIPS gcc compiler included in SimpleScalar toolset at optimization level –O2. The memory architecture we used for our experiments was separate level-1 instruction and data cache together with a unified level-2 cache and external memory. The processor clock speed was set at 650 MHz in all the simulations with 90 nm process power model. The energy model for the off-chip memory includes the memory and communication energy consumption.

In our previous work [3] we showed that video encoding applications are less sensitive to memory latency and a standard multilevel cache hierarchy provides performance that is almost as good as using 512 KB of dedicated on-chip zero-wait-state memory. The cache configuration used in this paper is the one we proposed in or study, 16 KB instruction, 32 KB data level-1 cache as well as 128 KB level-2 cache. The off chip memory was modeled as bursted memory, with first chunk latencies of 45 cycles and 5 cycles for the subsequent accesses.

The superscalar architecture is based on MIPS-IV architecture with six pipeline stages fetch, dispatch, issue writeback, and load/store queue refresh. The architecture supports out-of-order issue and execution by using Register Update Unit (RUU) [13] working as a reservation station for reordering and register-renaming of pending instructions. A Load/Store Queue (LSQ) is also used for employing a support for speculative execution. The results are stored in a store queue while pending resolve of execution. Table 1 shows the configuration of the superscalar processor.

Table 1. Superscalar processor configuration.

	2-way	**4-way**	**8-way**	**16-way**
Fetch,Decode,Issue,Commit Width	2	4	8	16
RUU size (inst.)	8,16,32,64, 128	8,16,32,64, 128	8,16,32,64, 128	8,16,32,64, 128
LSQ size (inst.)	8,16,32,64	8,16,32,64	8,16,32,64	8,16,32,64
Number Func. Unit	2	4	8	16

The configuration chosen for the H.264 encoder has screen resolution CIF (352x288) and four reference frames were used. The screen resolution has a linear impact on the performance, which we showed in our previous work [3].

Test sequences chosen in our experiments were foreman, mobile and news [9]. The main difference between these sequences is the amount of processing to encode the sequences. The mobile sequence is the most demanding sequence in terms of processing while news is least demanding. The foreman test sequence represents a good average

between these two sequences and therefore we have reported simulation results for this test sequence in this paper. In order to measure the overall performance of the system we used frames per second, which in our case is more relevant as we are performing video encoding. As we are dealing with handheld battery driven embedded devises we use total energy consumption rather than power consumption. For evaluating ILP we used instructions per cycle (IPC).

4.5 Experimental Results and Discussion

In this section, we present evaluation of the superscalar architecture when performing video encoding. In this section, due to space limitation, we only present the results from foreman test sequence at CIF screen resolution, since it provides a good average evaluation. In sub-section A, we measure the amount of ILP and the impact it has on DLP in terms of percentage improvement for SIMD as well NonSIMD architecture. In sub-section B, we present the impact of ILP on performance. Sub-section C deals with the energy consumption of the architecture. Finally, we discuss experimental results and their implications in sub-section D.

4.5.1 Amount of ILP in H.264 and impact on DLP

To observe the amount of ILP in the H.264 we evaluated IPC for a set of different superscalar width (fig. 3), and observed first that in-order architecture does not give any improvement despite increased processor width. This is due to inability to better schedule instructions when data and resource dependencies exist. To some extent better compiler can solve this, but there are still some dependencies that can only be handled during runtime, such as conditional branches. In the case of out-of-order (OoO), we observe an IPC improvement up to 8-way superscalar processor. This is quite natural since the OoO runtime scheduler can much better utilize the added hardware.

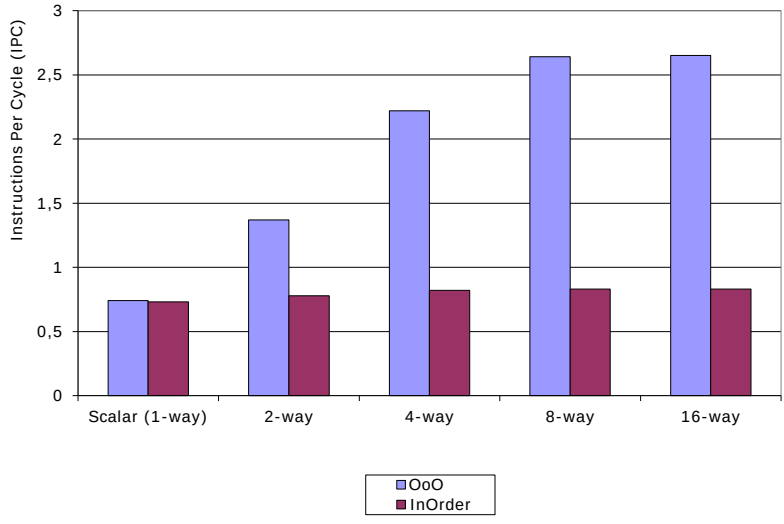


Figure 3. Instructions per cycle (IPC) for in-order vs OoO with different superscalar issue width.

Figure 4 shows the percentage improvement for SIMD and NonSIMD architecture, when increasing the width of the processor, using conventional ILP technique. This percentage improvement is measured when going, from scalar to 2-way, from 2-way to 4-way, etc. Our results show that in the case of SIMD extended processor already when increasing to 2-way superscalar we have best improvement and utilization of the system. This occurs due to the ability of better scheduling of SIMD instruction, as these instructions are independent of each other. By providing a 2-way issue capability, SIMD unit utilization increases. Going beyond 2-way does not provides more utilization as we have saturated SIMD utilization. For the NonSIMD architecture, this occurs at 4-way superscalar. The reason is that a conventional ILP can better schedule instructions, solve data dependencies as well as handle resource limitations. This shows the distance between two independent instructions is often larger than two instructions. We observed highest percentage improvement at 4-way, going beyond this we observe the same saturation, and the improvement become much smaller.

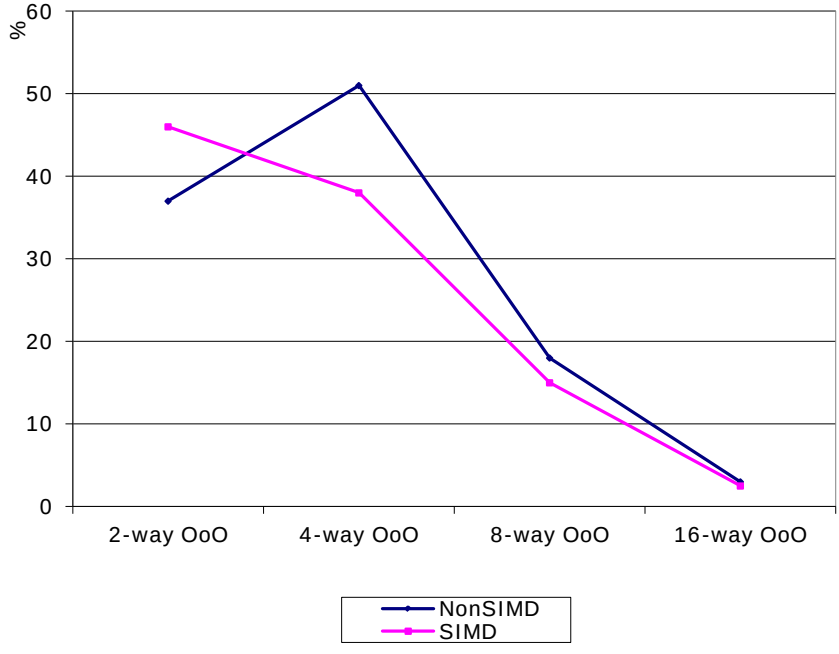


Figure 4. Percentage improvement of SIMD and NonSIMD architecture when going from scalar to 2-way, from 2-way to 4-way, from 4-way to 8-way and from 8-way to 16-way.

4.5.2 Encoding performance by combining ILP and DLP

The performance target was to encode 30 Frames/s at CIF screen resolution. Looking at fig. 5 this was almost achieved by the 2-ways SIMD extended architecture. For achieving the 30 frames/s target in the NonSIMD architecture, we need to have at least 4-way superscalar. We also observe linear improvement of performance with added issue width up to 8-way. Beyond this point, the improvement is nonexistent, as the potential ILP is exhausted. Fig. 5 depicts performance, measured in frames/s, of different analyzed architectures. The data presented in this figure illustrate importance of OoO execution. It can be explained by the fact that most ILP can be found between iterations rather than within a single iteration. The in-order processor improvements are negligible. This happens even though the in-order processor dynamically schedules instructions and has non-stalling pipeline when memory stalls.

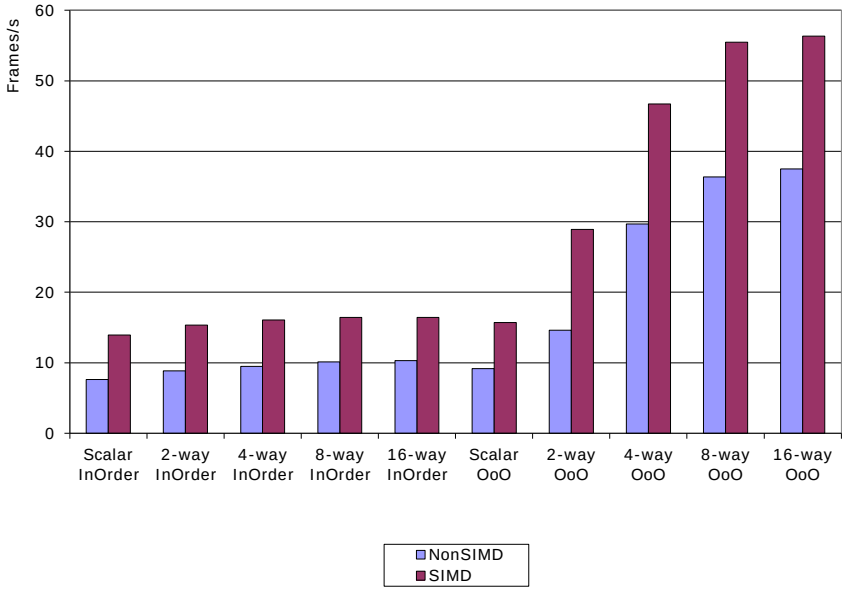


Figure 5. Different architectures performing H.264 encoding, In-order scalar, in-order superscalar, out-of-order (OoO) scalar and OoO superscalar.

4.5.3 Hardware and energy costs

The superscalar OoO implantation in our architecture is based on a central instruction window implemented as Register Updata Unit (RUU) and a Load/Store Queue (LSQ). Fig. 6 illustrates the performance of the encoder while increasing the RUU and LSQ sizes. We observe that already at 16 instructions wide, RUU and LSQ we have achieved a breaking point of improvement. One drawback with RUU, discussed in [10], is the sensitivity of the design for small instructions windows compared to using a more complex stack based or a reorder buffer central window.

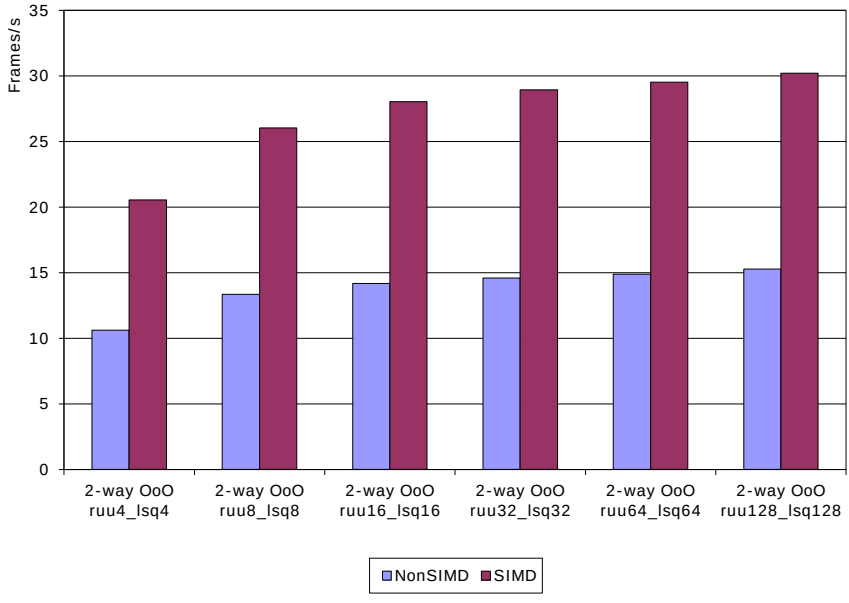


Figure 6. The Central instruction window size. The RUU instruction queue size has non-linear performance degradation when the size of the queue is below 8 instructions.

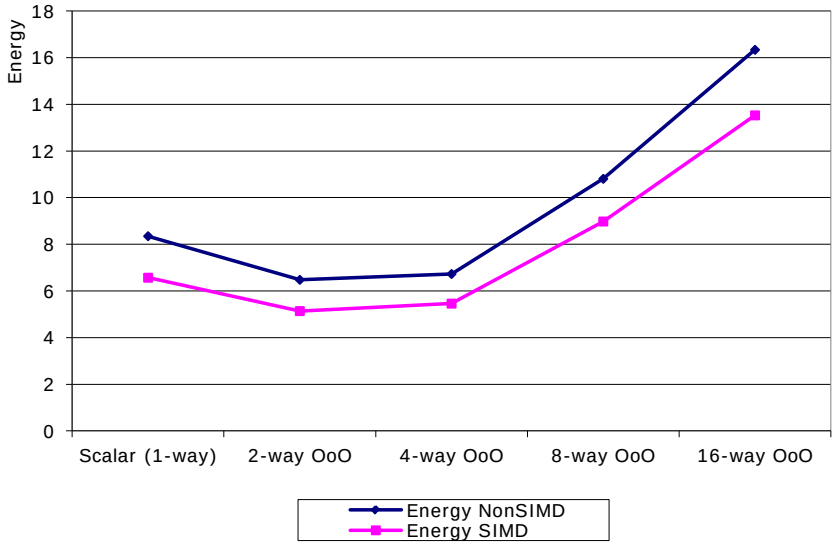


Figure 7. Performance and energy consumption for an out-of-order (OoO) with increased processor width.

Energy consumption depicted in fig. 7 indicates a minimum at 2-way for the SIMD as well as NonSIMD. The architecture operates at the optimal point of energy when both added hardware cost in terms of energy and performance gain are considered. Fig.8 illustrates the overall energy consumption of in-order and OoO superscalar designs. We observe higher serge in energy for the in-order architecture because the added hardware is underutilized. The in-order design has slightly less energy consumption in the scalar case as the performance of added OoO for the scalar processor is negligible, but the performance gains become more apparent as the processor width increase, showing more favorable energy consumption.

4.5.4 Discussion

Instruction level parallelism has a positive impact on the performance of video encoding application. This impact provides the necessary performance to achieve the target of encoding 30 frames/s at CIF screen resolution. For the SIMD extended architecture this is achieved already at 2-way superscalar. We have also the best improvement for this architecture as well as lowest energy consumption. For the NonSIMD architecture, this occurs at 4-ways superscalar. One important observation is the dynamic scheduling of instructions, which in our architecture is done by a central

instruction window design. The alternative to this would be using static scheduling and during compile time schedule instructions. The added OoO hardware is quite limited, both in terms of processor width as well as in terms of size of RUU and LSQ.

4.6 Related work

In [1], the authors propose a vector SIMD-VLIW architecture where they show that for MPEG-2 encoding they can achieve overall 1.5X speedup when both DLP and ILP regions are taken into account, even though they increase the width of their architecture from 2-ways to 8-ways. They also show that their vector approach improves their overall average performance for a set of applications. In our study, we evaluate a SIMD extended superscalar architecture and focus on H.264 encoder, which has more ILP than previous standards like MPEG-2.

The authors of [14] focus on different media benchmark kernels and applications, such as DCT, motion estimation kernel, speech, and jpeg encoding applications. They use the results from their evaluations to propose architecture for the media applications they have selected. They report good SIMD utilization for their applications but there is a need to take care of parallelism existing outside main loops and kernels. In their conclusions, they state that conventional ILP techniques need at least 8 or 16-way superscalar processor to provide improvements. We observe that in the case of H.264 video encoding, already at 2 and 4-way superscalar architecture, there is sufficient improvements combined with SIMD extension.

In [15], the authors evaluate MediaBench II video, which includes among others H.264 encoder. They evaluate the different applications on 8-ways VLIW, in-order and out-of-order architectures. They show in their work that an 8-issue VILW and an 8-issue in-order superscalar processor perform equally well with a modest IPC of around 1.3 for the H.264 encoder, even though they have used different compiler optimization techniques. In the case of 8-issue out-of-order superscalar processor, the achieved IPC was around two for the H.264 encoder. Our work complements their work by combing the ILP together with DLP using a SIMD unit, and evaluates the overall performance when combining the two approaches.

4.7 Conclusions

This paper analyzes video encoding application H.264 in terms of ILP as well as DLP. The results show the SIMD extended processor already at 2-way superscalar OoO have sufficient gains for meeting the performance requirements of encoding 30 frames/s at CIF screen resolution. Out-of-order scheduling is extremely important in video encoding since parallelism found in modern encoders is not limited to data parallelism and parallelization of loops does not provide enough performance improvement.

References

[1] E.Salami and M.Valero, "A Vector-μSIMD-VLIW Architecture for Multimedia Applications," in Proceedings IEEE ICPP'05, 2005.

[2] A.R..Iranpour and K.Kuchcinski, "Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors," in Proceedings SAMOS VI workshop: Embedded Computer Systems: Architectures, MOdeling, and Simulation Samos, Greece, July 17-20, 2006.

[3] A.R..Iranpour and K.Kuchcinski, "Evaluation of SIMD Architecture Enhancement in Embedded Processors for MPEG-4," in Proceedings IEEE DSD'04, Sep. 2004.

[4] 3GPP specification release 6, www.3gpp.org

[5] V.Lappalainen, A.Hallapuro and T.D.Hamalainen, "Performance of H.26L Video Encoder on General-Purpose Processor," Kluwer Journal of VLSI Sig. Proc., Vol. 34, No. 3, pp. 239-249, 2003.

[6] Joint Video Team (JVT) of ISO/IEC MPEG, ITU-T VCEG "Text of ISO/IEC 14496 10:2004 Advance Video Coding Standard (second edition)," ISO/IEC JTC1/SC29/WGII/N6359, Munich, Germany, March 2004.

[7] H.264/AVCSoftwareCoordination,JM,http://iphome.hhi.de/suehring/tml

[8] C.Priddle, "H.264 video encoder optimization with focus on very low complexity algorithms," M.S. thesis, Uppsala University, April 2005.

[9] Test sequences, http://www.chiariglione.org/mpeg/

[10] M.Johnson, "SuperScalar Microprocessor Design", Prentice Hall Series in Innovative tech., ISBN 0-13875634-1, 1991.

[11] T.Austin, E.Larsen and D.Ernst, "SimpleScalar: An infrastructure for computer system modeling," IEEE Computer, Vol. 35, Issue 2, pp. 59-67, Feb. 2002.

[12] D.Brooks, V.Tiwari and M.Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in Proceedings ISCA'00, pp. 83-94, June 2000.

[13] J.E.Smith and G.S.Sohi, "The Microarchitecture of Superscalar Processors", in Proceedings IEEE, Vol. 83, No. 12, Dec. 1995.

[14] D.Talla, L.K.John and D.Burger, "Bottlenecks in Multimedia Processing with SIMD Style Extension and Architectural Enhancements," IEEE Trans. Computers, Vol. 52, No. 8, pp. 1015-1031, Aug. 2003.

[15] J.E.Fritts, F.W.Steiling and J.A.Tucek, "MediaBench II: Expediting the next generation of video systems research," in Embedded Processors for Multimedia and Communications II, Vol.5683, No.5683 pp.79.93, editors, S.Sudharsanan, V.M.BoveJr. and S.Panchanathan ISBN/ISSN:0-8194-5656-X,Mar.2005.

Paper V

Design Space Exploration for Optimal Memory Mapping of Data and Instructions in Multimedia Applications to Scratch-Pad Memories[5]

Abstract. *In this paper, we propose a new methodology for optimal memory mapping of data and instructions to Scratch-Pad Memories (SPM). In the mapping process, we optimize, as the first priority, the number of memory accesses to minimize power consumption. Minimization of external memory accesses lowers switching activity and therefore power consumption. The optimization is done by finding Pareto points, using multi-objective optimization that combines different cost functions. Our methodology is intended to be used in real-life situations in industry where there is often a need for mapping third party applications to a specific architecture. For evaluating our methodology, we also use commercial video H.264 and audio eAAC+ applications. Our experiments show that SPM is well suited for these applications for reducing external accesses to reduce power consumption but has limited significance on overall performance improvements. The proposed methodology provides a way to combine SPMs with caches to optimally use this memory architecture. Our experiments indicate high accuracy of our methodology for predicting SPM and external memory accesses. We have obtained 90% accuracy between results of our methodology and results for executing applications on a given architecture.*

[5] This paper is a reformatted version of *Design Space Exploration for Optimal Memory Mapping of Data and Instructions in Multimedia Applications to Scratch-Pad Memories*, in Proc. of 7th ESTIMedia 2009, Grenoble, France, October 15-16, 2009.

5.1 Introduction

There has always been a drive to find the most optimal solutions in embedded resource-constrained systems in the past. But as time has passed the complexity of such systems has increased, both in terms of overall architecture, but also in terms of application sets run on these systems. The embedded systems share now in many ways the same complexity as other systems, such as high performance systems. This is especially true for processing elements as well as memory hierarchy. How these subsystems are utilized is becoming very important and a key area is memory utilization. The memory architecture is now a mix of different memories, such as cache memory hierarchy and software controlled Scratch-Pad Memories (SPM).

At the same time the importance of being able to handle different applications with high computational intensity and complexity has become a key issue for handheld embedded devices, such as mobile phones and PDAs. One such category of applications is multimedia, such as video and audio applications. These applications have often very different execution profiles. They also put huge demands on the entire system, processor, memory, buses and other parts. Moreover, modern video and audio applications, such as H.264 and enhanced Advanced Audio Codec plus (eAAC+), utilize optimized algorithms that have totally different execution profile than standard reference codecs used previously in many studies.

Looking at the state-of-the-art memory architectures in embedded systems, the most typical architectures consist of a cache based memory, which is often combined with software controlled zero-wait-state memory, such as Scratch-Pad Memories (SPM). This is a good combination, since SPM and caches have very different behavior. SPMs are optimal for regular access patterns, but not for runtime dependent behavior. Caches on the other hand are well suited for this. There is a need for both, but main issue is, how to optimally use this combined memory architecture.

There exists a need for a method that systematically finds the optimal memory mapping for different application. As this is not a trivial problem, we need to perform design space exploration. There could be different criteria for optimization, for example, improving performance by lowering the execution time or lowering power consumption by lowering access to an external memory. The latter is what we have chosen to focus our attention on, as external accesses are one of the main contributing factors on the overall power consumption for application execution.

In this paper, we propose a powerful methodology based on finding Pareto points for different memory configurations. The solution points provide optimal memory mappings for data and instructions in multimedia applications. In our approach, we optimize number of memory accesses, which can be viewed as optimizing switching activity α in the $P \cong \alpha C f V^2$ power equation. The optimization is done by finding Pareto points using up to four-dimensional cost functions.

We evaluate our method using proprietary video H.264 and audio eAAC+ applications not relying on kernels and standard benchmarks. This is important as the execution profile and resource requirements are very different for optimized commercial applications compared to standard benchmarks and test applications. This is essential in order to get correct and accurate behavior, when evaluating a design or a method, as it depends very much on the quality of the applications used. This is especially true for multimedia applications, such as video and audio codecs.

The remainder of this paper is organized as follows. In section 2 we discuss related work. The proposed method, which is used in our approach, is presented in section 3 and in section 4 we describe our experimental setup, the applications used and also we present and discuss our experimental results. In section 5 we give concluding remarks.

5.2 Related Work

There has been significant work done in the area of evaluating and proposing usage of SPM for various applications including audio and video codecs. Much work has focused on proposing architectures, which utilize SPM, by rewriting significant portions of applications to fit them to proposed architectures [15,12,1]. Other studies have looked at data mapping on SPM, both statically [2,4,13] and dynamically [16,3]. Common approach to these prior works has been mainly based on code transformations, i.e. loop and trace analysis and rewriting of applications for optimization. In our work we have worked with black box code model, as this is very often the case in industry when working with third party applications.

There has also been work done, where a combination of using compile time and runtime approaches, is proposed [11]. This is done by inserting custom instructions to inform hardware to control data placement. Another closely related architectural approach uses locked caches and controls placement of memory objects [7]. Yet another approach focuses on managing memory space for different application in a SPM only architecture and thus eliminating caches totally [17]. A common approach for most of these papers has been the usage of kernels and benchmarks. In our study, we use state-of-the-art audio and video commercial applications tailored for handheld devices in order to achieve realistic and accurate overall behavior. The methodology we are proposing is based on analysis of runtime behavior of a specific application. The gathered data is then used together with a set of constrains to perform design space exploration by finding the optimal mapping of memory objects based on different cost functions.

5.3 Our Approach

In this section, we present a method for optimally mapping data and instructions to Scratch-Pad Memories (SPM).

5.3.1 Overall method

Figure 1 depicts the overall design flow of our methodology. The target application is analyzed, either automatically or manually, to identify the important and critical candidate data, such as arrays, constants and code blocks, for example inner loops in functions, sequences of instructions or specific functions. Next, a mapping of the selected objects to Scratch-Pad Memory (SPM) is made.

Each object is individually placed in SPM and is simulated. Thus we perform multiple simulations depending on the number of candidate objects. For upper and lower bound we also perform simulations for architectures without SPM and a case where all objects are allocated to the SPM. SPM is used for data that is generated during execution, intermediate results, and is compile-time allocated. The reason for this is that commercial code, for mainly performance reasons, seldom uses run-time allocated data. SPM is also used for selected parts of program code, based on execution profile that lowers external accesses.

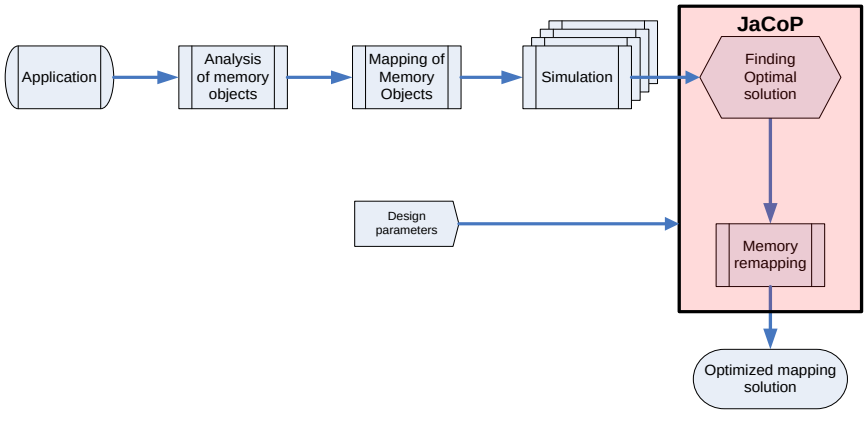


Figure 1. Design flow.

The remapped application is evaluated, using a cycle accurate simulator [19], on the target architecture. The simulation data for each individual memory object together with design parameters, such as SPM and cache sizes are used by cost functions to find optimal solutions. The optimization is performed by the constraint solver, JaCoP [14], and a set of solutions for different memory mappings and different SPM sizes is obtained. The optimal mapping is based on the Pareto points found by JaCoP for a specific cost function and SPM size. In our proposed method, we can optimize four different cost functions: maximizing SPM accesses, minimizing external memory accesses, cache accesses and minimizing execution cycles. To further enhance the

methodology, we perform two, three and four dimensional design space exploration using multi-objective optimization. This, by combining the cost functions mentioned above.

To validate our method we run simulations on selected memory mappings and our experiments indicate that the accuracy of predicting SPM accesses is 98%. The accuracy for predicting external accesses is around 90-95% and for predicting cache accesses is 80-85%. A reason for this drop in accuracy for caches and external memory accesses is due to the nature of caches. We make the assumption that accesses are not correlated if data is put in SPM, but this assumption is not valid for caches. The figures presented in the experimental section can be used for selecting the right configuration of memory. Basically, the figures show the results of design space exploration for obtaining optimal mapping of selected objects to SPM depending on different cost functions.

Our method is generic and can systematically be used to analyze and find an optimal memory mapping for a given application and architecture. The method performs multi-objective optimization in the design space without being limited to only single application. It can be used for multiple applications and other memory architectures.

5.3.2 Pareto point generation

JaCoP constrain solver [14] has been used in our approach to find optimal memory mappings for our applications. We want to explore different memory mappings alternatives and select the one that best suits given requirements. Design space exploration can therefore be defined as a process of finding different solutions that provide trade-offs between design parameters. It is usually achieved by doing a multi-objective optimization, meaning the optimization that simultaneously optimizes more than one cost function. In such cases there is not a single optimal solution but instead we have Pareto points.

To define multi-objective optimization more formally we assume that there exist several optimization criteria or cost functions. For minimization, we define Pareto optimal solution as vector x, if all other solution vectors y have a higher value for at least one of the cost functions, or have the same value for all cost functions. Each vector x that is Pareto optimal is also called Pareto point or non-dominated or non-inferior point since it does not exist any other point that is better in respect to at least one criterion.

The generation of Pareto points can be formalized using constraint programming. The idea is to define all constraints for a problem and then start searching with most relaxed constraints. When a solution is found additional constraints are imposed and possible dominated solutions are removed. The additional constraints cut the part of the search space that can only have dominated points. The pseudo-code for this algorithm is presented in Figure 2. The algorithm uses depth-first-search (DFS) to search for a solution. When a solution is found the algorithm imposes new constraints to cut possibility of finding dominant solutions and removes already found dominated solutions. Finally the algorithm returns the set of Pareto points.

vector pareto ($\mathcal{V}$, *criteria*)
paretoPoints ← **nil**

while DFS ($\mathcal{V}$) ≠ **nil**
    paretoPoints ← *paretoPoints* ∪ (val_0, …, val_n)
    impose $criteria_0 < val_0 \vee \cdots \vee criteria_n < val_n$
    remove points dominated by (val_0, …, val_n) from *paretoPoints*
return *paretoPoints*

Figure 2. The Pareto points generation algorithm.

5.4 Experiments and Evaluation

In this section we present our experimental setup, applications used and results from evaluation of our method. Our method generates all non dominated solutions (optimal Pareto points). We conclude the section with discussion on applications behavior and power consumption when using Scratch-Pad Memory (SPM).

In our study we used SoC Designer with cycle accurate ARM prime cell models [5] provided by ARM. The compiler used was, the ARM compiler in RealView Development Suite (RVDS 3.0). The target hardware architecture comprises an ARM processor core with separate 16 KB instruction and data L1 caches. It also contains either unified or separate Scratch-Pad Memory (SPM) for both instructions and data, referred to as tightly coupled memory (TCM) in ARM processors.

The applications used in our study are two proprietary multimedia applications, audio eAAC+ and video H.264 full codecs. Table 1 and 2 shows the selected memory objects, their names, object number and sizes, that are identified in the analysis stage, for potential placement in SPM.

Table 1&2. Table 1 shows H.264 codec memory objects and their sizes. Table 2 shows eAAC+ codec memory objects and their sizes.

Object Number	H.264	Total SPM size
	No Objects in SPM	0
	All Objects in SPM	64201
V1	c64enc	6027
V2	CodeIntra	16084
V3	CodeMacroblock	3632
V4	Init	3140
V5	Loopfilter	6340
V6	MotionEstimation	7190
V7	Putbits	9304
V8	Ratedistortion	4248
V9	Transform	5048
V10	Speed3	3188

Table1

Object Number	eAAC+	Total SPM size
	No Objects in SPM	0
	All Objects in SPM	49224
A1	SBRbuffer	32768
A2	SyntQMFfilter	5120
A3	AnaQMFfilter	2560
A4	QMFbuffer	304
A5	EnvCal	864
A6	Trans	1024
A7	Overlappbuffer	6144
A8	Prevframe	440

Table2

5.4.1 Enhanced Advanced Audio Codec plus (eAAC+)

The eAAC+, used in our research, consisted of Advanced Audio Codec (AAC), Spectral Band Replication (SBR) and Parametric Stereo (PS). The eAAC+, also referred to as aacPlus v2, is part of MPEG-4 standardization [8]. The underlying core codec is the well known MPEG AAC codec with a typical bit rate of 128 kbps.

Table 2 shows the list of memory objects that were selected during the analysis phase. These objects are individually simulated and the simulation data is gathered for optimization. In the case of eAAC+, the memory objects are all data objects using Data Scratch-Pad Memory (DSPM). Using our methodology we have performed two, three and four dimensional optimization. We first present solutions for 2 cost functions, SPM size vs. SPM accesses and SPM size vs. MeM accesses.

Figure 3 shows two dimensional Pareto points for SPM and memory accesses for different SPM sizes. For eAAC+ we observe significant SPM accesses increase when going from 4KB to 16KB SPM. A reason is the impact of object number2 (SyntQMFfilter), which has significant higher access/size ratio, compared to the other objects. Looking at the memory accesses we observe an almost linear reduction in accesses. We do not observe a direct impact on the memory accesses due to the increased SPM accesses. This indicates that the increased SPM accesses are mostly moving cache accesses to SPM. However, the gains in reduction of memory accesses are still significant and start to flatten at 64KB SPM.

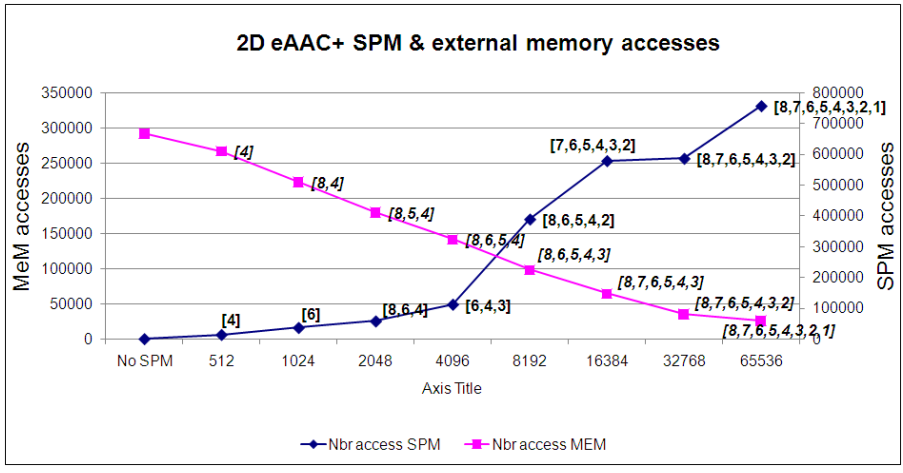


Figure 3. Two-dimensional Pareto point with the chosen memory objects, for external memory accesses with different SPM sizes when running eAAC+ codec.

As it can be observed the system finds different solutions for the same size of SPM optimizing memory accesses or SPM accesses. For example, at 2 KB SPM size, when external memory accesses are minimized the objects (8,5,4) are mapped to SPM. But for maximum SPM usage then objects (8,6,4) should be mapped. Our goal is to reduce the overall power consumption. This can be achieved by lowering number of external accesses. Moreover, we can reduce energy by decrease application execution time. For this reason we performed multi-objective optimization with SPM size, MeM accesses and Execution cycles.

Figure 4 shows a three dimensional Pareto points when optimizing execution cycles and memory accesses for different SPM sizes while running eAAC+ audio codec. In this case, we notice that for 1KB SPM, objects (8,4) should be mapped to the SPM. The figure can be used for selecting the right configuration of memory. We observe similar SPM and memory access behavior to that we saw for the two dimensional optimization. Due to presentation limitations we do not show a diagram for the four dimensional optimization, where we run a combined optimization between SPM size, execution cycles, memory and SPM accesses. In the eAAC+ we only utilize the DSPM. The reason for not utilizing the Instruction Scratch-Pad Memory (ISPM) is that the entire code fits into the cache. For an application, which is larger than the cache, we would allocate instruction code to ISPM in similar way as we have done for H.264.

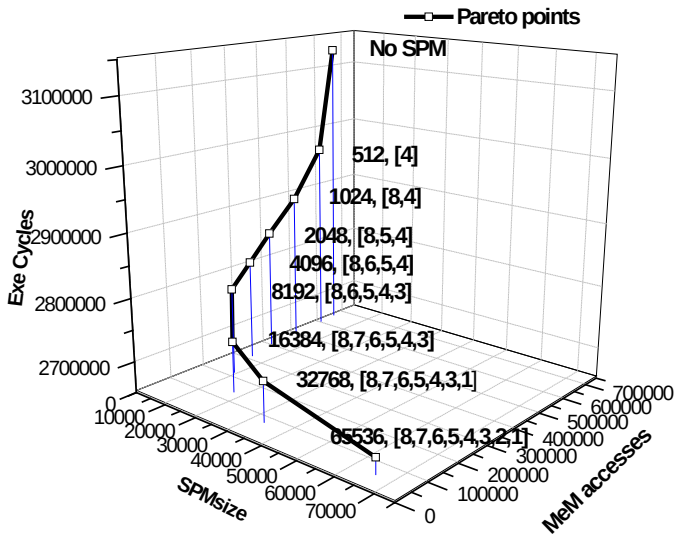


Figure 4. Three-dimensional Pareto point with the chosen memory objects, for external memory accesses and execution cycles with different sizes when running eAAC+ audio codec.

5.4.2 Advanced Video Codec H.264

The video standard used in our research is H.264. This standard is used in a variety of areas, such as videoconferencing, and it is also the main video standard recommended by 3GPP standardization group in release 6 [5]. The test sequence chosen in our experiments was *foreman* [20].

Our H.264 application implementation is proprietary software optimized for handheld devices. The H.264 implementation evaluated performs well against the reference implementation in terms of quality [6, 18]. The time complexity of our encoder is significantly lower and has a speedup of approximately 100 times with an average bit-rate increase of less than 20%, than the reference encoder. It uses optimized search mode and is configured for running different screen sizes.

For the H.264 encoder, we have observed, that the usage of DSPM has a very limited impact on external accesses as the usage of data is content dependent and thus we focus on ISPM. By using ISPM the external accesses are reduced up to 57%. Table 1 shows the list of objects that was selected during the analysis phase. Therefore in H.264 the focus is on instruction and not data for reducing external accesses (see Table 4). We use the same approach as we did in the case of eAAC+, two, three and four dimensional combined access optimization. Figure 5 shows the two dimensional Pareto points of SPM and memory accesses for different SPM sizes, when encoding the

foreman test sequence. We observe that there is not one-to-one correlation between SPM utilization and reduction of external accesses. An optimal usage of SPM does not automatically give us least external memory accesses. For example for 16KB SPM, the optimal mapping should be objects (9,5,3) if we maximize the SPM usage. But as we are more interested in lowering power consumption the mapping of objects (10,8,4,3) should be preferred.

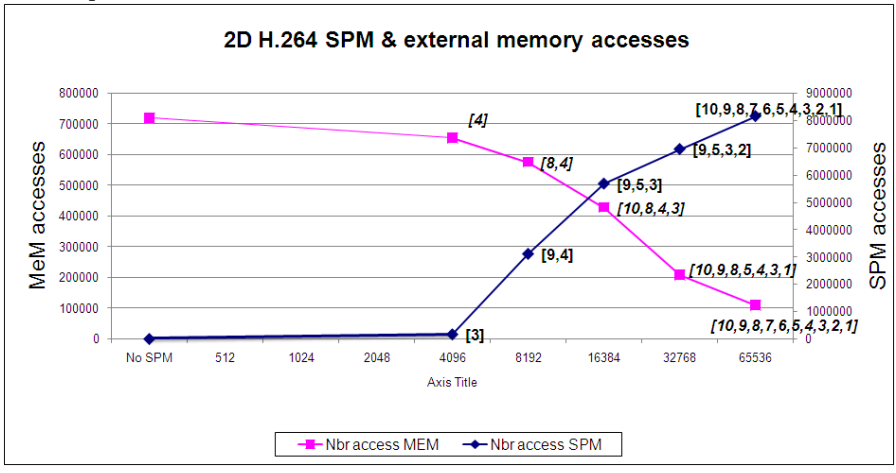


Figure 5. Two-dimensional Pareto point with the chosen memory objects, for SPM and external memory accesses with different sizes when running H.264 codec. As there are no memory objects smaller than 3140 bytes (see table1), we do not have any points for SPM sizes 512, 1024 and 2048 bytes.

Figure 6 shows a three dimensional Pareto points when combining execution cycles and memory accesses for different SPM sizes while running H.264 video codec. As can be seen in Figure 6, for 8, 16 and 32 KB there are more than one optimal Pareto point. These points are not dominated by each other, as we are optimizing in three dimensions and memory accesses and execution cycles do not dominates simultaneously. Depending on the desired optimization, execution cycles or memory accesses, we can either choose one or the other. In the case of H.264 the access pattern is not as in eAAC+ where certain selected objects stand out, making the Pareto points 2D curves and 3D surfaces much more linear when increasing the SPM size.

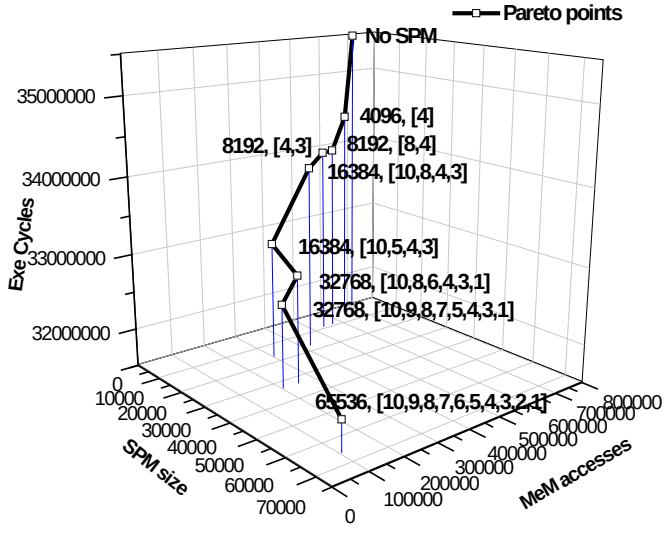


Figure 6. Three-dimensional Pareto point with the chosen memory objects, for external memory accesses and execution cycles with different sizes when running H.264 video codec.

5.4.3 Combined mapping of eACC+ and H.264 to a single unified SPM

When analyzing multimedia applications, audio and video, the combination of different applications on the overall system characteristics is an important issue. In this study, the focus has been on the impact of SPM on overall power due to external memory accesses. The question is when looking at both eAAC+ and H.264 in the same memory space, which of the memory objects, table 1 and 2, should be mapped to SPM. Our method not only can be used for partitioning of SPM for a single physical address space but can be used for mapping between different applications.

Figure 7, illustrates the overall impact on external memory accesses and SPM accesses for the combined optimized memory mapping. For example, for 64 KB SPM and the optimization of external memory accesses for our two applications, both of the largest objects A1 and V2 are not selected. This indicates that it is better to map smaller objects than to include the largest ones. But when SPM accesses are optimized, object V2 is selected instead of smaller objects. Conclusion from this is that object V2 moves cache accesses to SPM but has not significant impact on lowering external accesses.

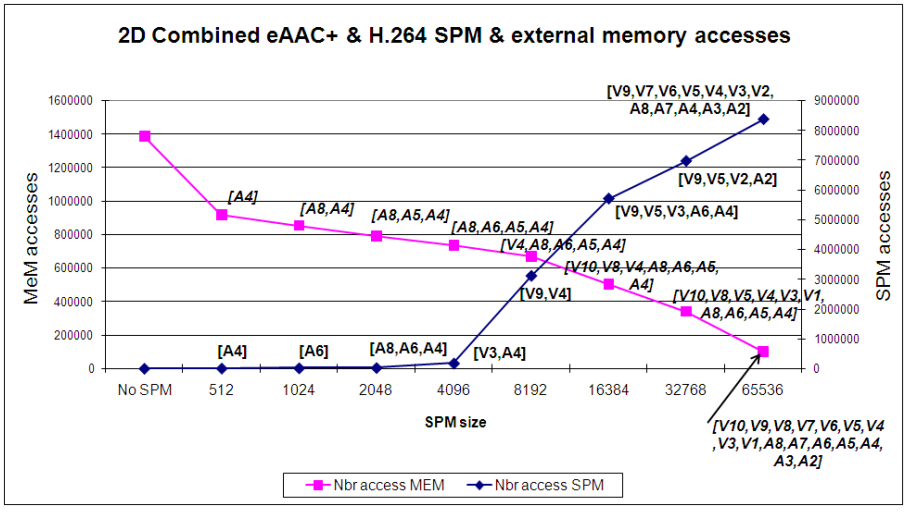


Figure 7. Two-dimensional Pareto point with the chosen memory objects, for external memory accesses with different SPM sizes when running eAAC+ and H.264 codecs mapped to a unified SPM.

Table 3. Shows Pareto points 1-16 for the combined eAAC+ and H.264 codecs in figure 8. For each optimization point the chosen objects, audio 'A' and video 'V', are shown.

Points	SPM size, [video 'V' and audio 'A' objects]
	No SPM
1	512,[A4]
2	1024,[A8,A4]
3	2048,[A8,A5,A4]
4	4096,[V4,A8,A4]
5	4096,[A8,A6,A5,A4]
6	8192,[V10,V4,A8,A5,A4]
7	8192,[V4,A8,A5,A4,A3]
8	16384,[V10,V8,V4,V3,A8,A5,A4]
9	16384,[V8,V4,V3,A8,A6,A5,A4,A3]
10	16384,[V10,V8,V4,A8,A6,A5,A4,A3]
11	32768,[V10,V9,V8,V5,V4,V3,V1,A8,A4]
12	32768,[V10,V9,V8,V6,V4,V3,A8,A6,A5,A4,A3]
13	32768,[V10,V8,V6,V4,V3,V1,A8,A6,A5,A4,A3]
14	32768,[V10,V8,V5,V4,V3,V1,A8,A6,,A5,A4,A3]
15	65536,[V10,V9,V8,V7,V6,V5,V4,V3,V2,V1,A8]
16	65536,[V10,V9,V8,V7,V6,V5,V4,V3,V1,A8,A7,A6,A6,A5,A4,A3,A2]

Figure 8 shows the three dimensional Pareto point optimization for the cost functions SPM size, Memory accesses and execution cycles, when mapping the combined, audio and video codecs, to the unified SPM. The Pareto points and their mapping configuration are presented in Table 3. For example for 16KB SPM there are three points 8, 9 and 10 with different configurations. None of these points dominated the others fully as we are optimizing with three cost functions. In this case the point with lowest external memory accesses is 10, as illustrated in Figure 8.

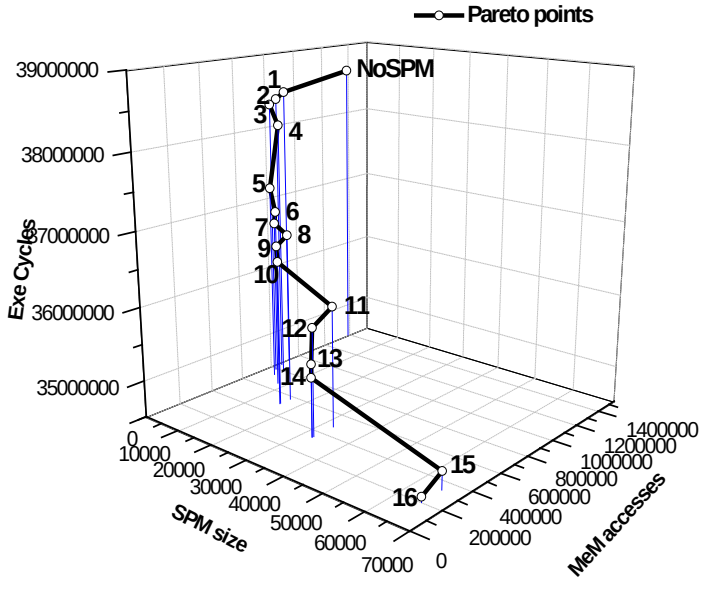


Figure 8. Three-dimensional Pareto point with the chosen memory objects, for external memory accesses and execution cycles with different sizes when combining eAAC+ audio and H.264 video codec.

5.4.4 SPM Impact on Power and Execution Time

There is significant power saving due to reduction in external accesses, but limited reduction in execution time. SPM are not and should not be used with the focus on reducing execution time for multimedia application, but rather used for increasing memory predictability as well as lowering total power consumption. Figure 9 and 10 show the impact of using SPM for reducing power consumption as we lower the external memory accesses. When looking at switching power consumption P ≅αCfV2 the capacitance C and the voltage V are the main factors. There are two factors one needs to consider, when analyzing power consumption due to remapping of memory

objects. The first is internal capacitance of the chip, which is significantly lower than external capacitance. The second is the chip internal voltage, typically 0.9V and the external IO voltage of around 1,8V. Putting it together one concludes that the main power consumption contributors are external accesses.

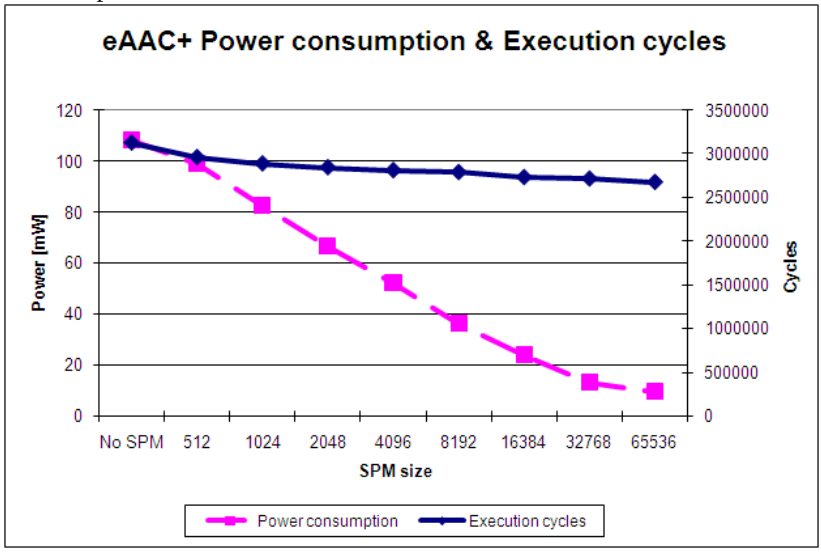


Figure 9. Impact of SPM usage on eAAC+ for power consumption and execution cycles.

At the same time we observe the very limited impact of SPM on the overall execution cycles. The impact of SPM for speeding up applications is limited as the SPM and caches are both zero-wait-state memories, thus giving negligible speed up. In the case of audio the intermediate results are best suited to be mapped to SPM. The instruction code gives no improvements as the entire code can fit in the cache. In the case of video the code is larger and there certain often accessed code is well suited to be mapped to SPM. When it comes to data in video applications there are a couple of factors which are involved and making SPM not suitable. Firstly, content dependent behavior of video applications limits the possibility of mapping the right data to SPM, since not all data in video applications is used. Secondly, there is difference in video and audio algorithms, where the video algorithms use less advance techniques and filters, making less re-usage of data. Thirdly, the amount of data that is allocated to memory is huge in video codecs compared to audio codecs, and the small size of DSPM, have very limited overall impact.

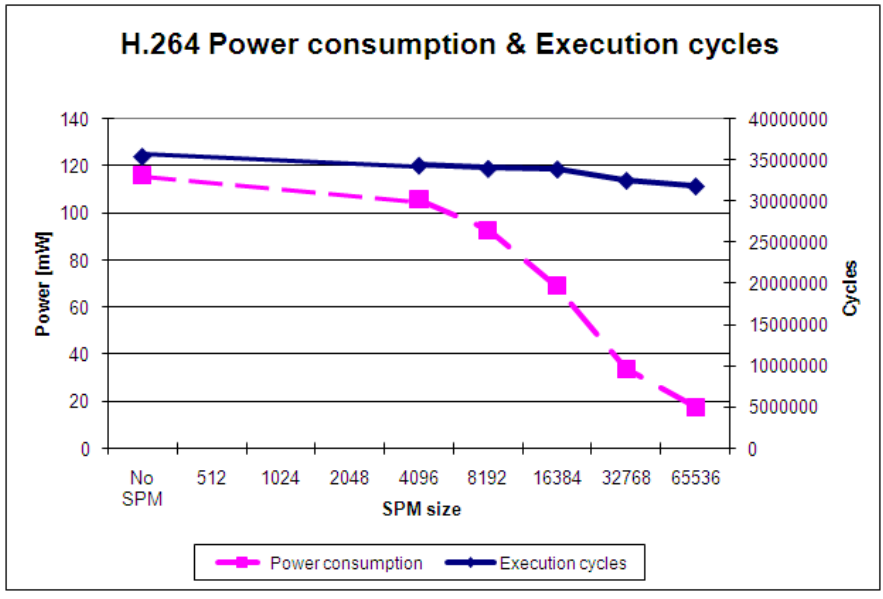


Figure 10. Impact of SPM usage on H.264 for power consumption and execution cycles. As there are no memory objects smaller than 3140 bytes (see table1), we do not have any points for SPM sizes 512, 1024 and 2048 bytes.

5.4.5 Instruction vs. Data in Audio and Video

There are significant differences between audio and video not only in terms of the amount of data, but also the way it is processed. Video applications, such as codecs, are content dependent and the result is that one does not know which data to put in a dedicated memory at compile time. Also as the amount of overall data is huge and at the same time very limited portions are used, putting all data in a dedicated memory neither realistic nor efficient. As we have shown in our previous work [9] the amount of gains are very limited.

Audio applications on the other hand are more algorithm dependent and all input data is processed in a predictable way, making it ideal for using dedicated memory, such as SPM. All input data is processed in a predefined algorithm regardless of the content. This enables us to identify and remapping the data types that has beneficial impact on lowering external accesses.

This difference is illustrated in table 4, where we see that the introduction of DSPM for data has no impact on the number of external accesses in the case of video codec H.264. The contrary could be said about the impact of ISPM for code and instructions, where we observe significant impact on the video applications. One reason being that compared to previous video codecs, such as MPEG-4, H.264 has much control code.

We have also shown this in our previous work [10], where impact of instruction handling is essential in gaining the performance through the usage of instruction level parallelism (ILP). The similar beneficial impact of handling instruction code, we observe when using ISPM for instructions resulting in lowering external memory accesses.

The opposite can be said about audio codec eAAC+. There the introduction of DSPM has significant positive impact on the external accesses, as shown in our experiments. But for instruction code ISPM has negligible impact, as our audio application fits into the cache. As pointed out earlier, if our audio application would not have fit entirely in the cache then we would have allocated selected part to the ISPM.

5.5 Conclusions

In this paper, we propose a generic methodology that explores the design space by determining Pareto point for different combined cost functions. The method systematically analyzes and finds optimal memory mapping for a given application and architecture. By using this method there is only a need to perform limited number of simulations, number instead of performing extensive simulation of all possible combinations. The applications we have used in our study are two proprietary state-of-the-art multimedia applications eAAC+ audio codec and H.264 video codec. In this study, we have focused on reducing external memory accesses and at the same time increasing the utilization of Scratch-Pad Memory (SPM). For this purpose we have looked both at data as well as instructions. Among the findings we have made are that SPM is good for reducing external accesses to reduce power consumption, but has limited significance on overall performance improvements. Our experiments indicate high accuracy for predicting SPM and memory accesses, above 90%. The methodology presented here can be adapted quite easily to real industrial situations where there is often a need for mapping third party application to a specific architecture.

References

[1] M. J. Absar, P. Marchal and F. Catthoor, “Data-Access Optimization of Embedded Systems Through Selective Inlining Transformation”, in Proceedings ESTMED, Sept. 22-23, 2005.

[2] M.J.Absar and F. Catthoor, “Analysis of Scratch-Pad and Data Cache performance Using Statistical Methods”, in Proceedings ASPDAC’06, Jan 24-27, 2006.

[3] D. Atienze, et al., “Efficient system-level prototyping of power-aware dynamic memory managers for embedded systems”, INTEGRATION the VLSI Journal 39, pp. 113-130, 2004.

[4] A. Dominguez, S. Udayakumaran and R. Barua, “Heap Data Allocation to Scratch-Pad Memory in Embedded Systems”, in Journal of Embedded Computing, Vol. 1, Issue 4, Dec, 2005.

[5] 3GPP specification release 6, www.3gpp.org

[6] H.264/AVCSoftwareCoordination,JM,http://iphome.hhi.de/suehring/tml

[7] E. G. Hallnor and S. K. Reinhardt, “A fully associative software-managed cache design”, in Proceedings ISCA’00, pp. 107-118, May, 2000.

[8] HE AAC version 2, “MPEG-4 in document ISO/IEC 14496-3:2005 (with 14496-3:2005/Amd.2. for HE-AAC v2”, 2005.

[9] A.R..Iranpour and K.Kuchcinski, “Memory Architecture Evaluation for Video Encoding on Enhanced Embedded Processors,” in Proceedings SAMOS VI workshop: Embedded Computer Systems: Architectures, MOdeling, and Simulation Samos, Greece, July 17-20, 2006.

[10] A.R..Iranpour and K.Kuchcinski, “Performance Improvement for H 264 Video Encoding using ILP Embedded Processor”, in Proceedings 9th Euromicro Conference on Digital SystemDesign, Cavtat/Dubrovnik, Croatia, August 30th - September 1st, 2006.

[11] A. Janapsatya, A. Ignjatovic and S. Parameswaran, “Hardware/Software Managed Scratchpad Memory for Embedded Systems”, in Proceedings ICCAD, pp. 370-377, 2004.

[12] A.Janapsatya, A. Ignjatovic and S. Parameswaran, “A Novel Instruction Scratchpad Memory Optimization Method based on Concomitance Metric”, in Proceedings ASPDAC’06, Jan 24-27, 2006.

[13] M. Kandemir, et al., “A Compiler-Based Approach for Dynamically Managing Scratch-Pad Memories in Embedded Systems”, IEEE Trans. Computer aided design of Integrated Circuits and Systems, Vol. 23, NO.2, Feb. 2004.

[14] Krzysztof Kuchcinski, “Constraints-Driven Scheduling and Resource Assignment”, in ACM Transactions on Design Automation of Electronic Systems (TODAES), 8(3), pp. 355-383, 2003.

[15] C. Kulkarni, et al., “Cache Conscious Data Layout Organization for Embedded Multimedia Applications”, in Proceedings DATE, pp. 686-693, 2001.

[16] S. Mamagakakis, et al., “Custom Design of Multi-level Dynamic Memory Management Subsystem for Embedded Systems”, in Proceedings Signal Processing Systems, pp. 170-175, Oct, 2004.

[17] O. Ozturk, M. Kandemir and I. Koleu, “Shared Scratch-Pad Memory Space Management”, in Proceedings 7th ISQED’06, 2006.

[18] C.Priddle, “H.264 video encoder optimization with focus on very low complexity algorithms”, M.S. thesis, Uppsala University, April 2005.

[19] SoC Designer & RealView Development Suite (RVDS 3.0) , ARM, http://www.arm.com/ and Carbon Design Systems, http://carbondesignsystems.com/

[20] Test sequences, http://www.chiariglione.org/mpeg/

[21] F. Rossi, P. Van Beek and T. Walsh (EDS.), “Handbook of Constraint Programming”, Elsevier, 2006.