

Loughborough University Institutional Repository

Parallelism and the software-hardware interface in embedded systems

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: <https://dspace.lboro.ac.uk/2134/14467>

Publisher: © V. A. Chouliaras

Please cite the published version.

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (<https://dspace.lboro.ac.uk/>) under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>



University Library

Author/Filing Title CHOWLIARAS, V.

Class Mark T

Please note that fines are charged on ALL
overdue items.

FOR REFERENCE ONLY

0403191653



Parallelism and the Software-Hardware Interface
In
Embedded Systems


By

Vassilios Apostolos Chouliaras

A Doctoral Thesis submitted in partial fulfilment of the requirements for the award of Doctor
of Philosophy of Loughborough University

June 2005

© V. A. Chouliaras

 Loughborough University Pilkington Library
Date JAN 2006
Class T
Acc No. 0403191653

ACKNOWLEDGMENTS

I would deeply like to thank my wife, Nantia, for continuous support throughout the long process of conquering the Knowledge. Her support, understanding and Love, kept me focused and composed throughout the years of working professionally away from the Mediterranean. It was her who decided that I should accept the academic post in the Midlands, instead of the industrial post in downtown San Jose and I have to admit, three years after that extremely difficult decision, that she was right and I was wrong.

I deeply acknowledge the support of my family in Athens throughout all my 35 and half years. Without their clear vision, and persistence in pursuing the Knowledge, I would have never been able to understand the ways our world is working.

I gratefully acknowledge the support of my colleague, Dr. Jose Luis Nunez-Yanez. Jose was the sole microelectronics engineering expert I could rely on for bringing our research to life and the accomplishments and success of the microelectronics research group were also due to his hard work and dedication.

I most sincerely thank Dr. David Mulvaney for being always available, always technically sound, always listening and giving solution to most problems, trivial or otherwise. David provided much assistance in the group and was an excellent academic probation adviser.

Finally, I would like to sincerely thank all my academic colleagues and friends here in Loughborough. In particular, I would like to mention Professor Roger Goodall for being an academic raw model to me and one that I simply am unable to reach.

This Thesis is dedicated to the Memory of my Dear Friend,

Dimitrios Daniolos

Who left us so suddenly

12 June 1969 – 11 June 2004

ABSTRACT

This thesis by publications addresses issues in the architecture and microarchitecture of next generation, high performance streaming Systems-on-Chip through quantifying the most important forms of parallelism in current and emerging embedded system workloads.

The work consists of three major research tracks, relating to data level parallelism, thread level parallelism and the software-hardware interface which together reflect the research interests of the author as they have been formed in the last nine years.

Published works confirm that parallelism at the data level is widely accepted as the most important performance leverage for the efficient execution of embedded media and telecom applications and has been exploited via a number of approaches the most efficient being vector/SIMD architectures. A further, complementary and substantial form of parallelism exists at the thread level but this has not been researched to the same extent in the context of embedded workloads. For the efficient execution of such applications, exploitation of both forms of parallelism is of paramount importance. This calls for a new architectural approach in the software-hardware interface as its rigidity, manifested in all desktop-based and the majority of embedded CPU's, directly affects the performance of vectorized, threaded codes.

The author advocates a holistic, mature approach where parallelism is extracted via automatic means while at the same time, the traditionally rigid hardware-software interface is optimized to match the temporal and spatial behaviour of the embedded workload. This ultimate goal calls for the precise study of these forms of parallelism for a number of applications executing on theoretical models such as instruction set simulators and parallel RAM machines as well as the development of highly parametric microarchitectural frameworks to encapsulate that functionality.

Table of Contents

ACKNOWLEDGMENTS.....4

ABSTRACT.....6

TABLE OF CONTENTS7

CHAPTER 1.....10

1.1 THEMATIC AREAS11

1.1.1 Data Level Parallelism11

1.1.2 Thread Level Parallelism.....13

1.1.3 The software-hardware interface.....14

1.1.4 Instruction Level Parallelism.....14

1.2 METHODOLOGY.....15

1.3 FURTHER WORK.....16

1.3.1 Advanced software optimization tools16

1.3.2 The System Design Framework.....17

1.3.3 Fault-Tolerant Parametric Multiprocessor Kernel for mission-critical applications.....18

1.3.4 The Ultimate ASIC CPU approach: The SS_SPARC.....19

1.4 AUTHOR CONTRIBUTIONS IN THE PUBLISHED WORKS.....19

Paper PJ1.....19

Paper PJ2.....19

Paper PJ3.....20

Paper PJ4.....20

Paper PJ5.....20

Paper PC1.....21

Paper PC2.....21

Paper PC3.....21

Paper PC4.....22

Paper PC5.....22

Paper PC6.....22

Paper PC7.....23

Paper PC8.....23

Paper PC9.....23

GROUP PJ: PUBLISHED JOURNAL PAPERS ON PARALLELISM AND THE SOFTWARE/HARDWARE INTERFACE IN EMBEDDED SYSTEMS.....24

PAPER PJ1: V. A. CHOULIARAS, J. L. NUNEZ, 'SCALAR COPROCESSORS FOR ACCELERATING THE G723.1 AND G729A SPEECH CODERS', IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, VOL. 49, ISSUE 3, AUG. 2003, PG. 703-71026

PAPER PJ2: V. A. CHOULIARAS, J. L. NUNEZ, K. KOUTSOMYTI, S. R. PARR, D. J. MULVANEY, S. DATTA, 'ON THE DEVELOPMENT OF A CUSTOM VECTOR ACCELERATOR FOR HIGH-PERFORMANCE SPEECH CODING', IEE ELECTRONIC LETTERS, VOL. 40, ISSUE 24, 25 NOV. 2004, PG 1559-1561	35
PAPER PJ3: V. A. CHOULIARAS J. L. NUNEZ, D. J. MULVANEY, F. ROVATI, D. ALFONSO, 'A MULTI-STANDARD VIDEO CODING ACCELERATOR BASED ON A VECTOR ARCHITECTURE', IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, VOL. 51, ISSUE 1, FEB 2005, PG 160-167	49
PAPER PJ4: J. L. NUNEZ, V. A. CHOULIARAS, 'HIGH PERFORMANCE ARITHMETIC CODING VLSI MACRO FOR THE H264 VIDEO COMPRESSION STANDARD', IEEE TRANSACTIONS ON CONSUMER ELECTRONICS VOL. 51, ISSUE 1, FEB 2005, PG 144-151.....	58
PAPER PJ5: GRECOS, C., SAPARON, A. AND CHOULIARAS, V., 'THREE NOVEL LOW COMPLEXITY SCANNING ORDERS FOR MPEG-2 FULL SEARCH MOTION ESTIMATION', REAL TIME IMAGING, 10, FEBRUARY 2004, PP 53-65	67

GROUP PC: PUBLISHED NATIONAL AND INTERNATIONAL CONFERENCE PAPERS ON PARALLELISM AND THE SOFTWARE-HARDWARE INTERFACE IN EMBEDDED SYSTEMS.....81

PAPER PC1: V. A. CHOULIARAS, J. L. NUNEZ, 'A SCALAR COPROCESSOR FOR ACCELERATING THE G723.1 AND G729A SPEECH CODERS', PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS (ICCE 2003), LOS ANGELES, CALIFORNIA, USA, ISBN:0-7803-8838-0	86
PAPER PC2: V. A. CHOULIARAS, J. L. NUNEZ-YANEZ, S. AGHA, 'SILICON IMPLEMENTATION OF A PARAMETRIC VECTOR DATAPATH FOR REAL-TIME MPEG2 ENCODING', PROCEEDINGS OF THE IASTED (SIP) 2004, HONOLULU, HAWAII, USA, ISBN: 0-88986-442-X.....	89
PAPER PC3: V. A. CHOULIARAS, J. L. NUNEZ, FABRIZIO. S. ROVATI, DANIELE ALFONSO 'A MULTI-STANDARD VIDEO CODING ACCELERATOR BASED ON A VECTOR ARCHITECTURE', PROCEEDINGS OF THE IEEE INTERNATIONAL CONFERENCE IN CONSUMER ELECTRONICS (ICCE 2005), LAS VEGAS, NEVADA, USA, ISBN: 07803-8839-9.96	
PAPER PC4: V. A. CHOULIARAS, J. A. FLINT, Y. LI, 'PARAMETRIC DATA-PARALLEL ARCHITECTURES FOR TLM ACCELERATION', PROCEEDINGS OF THE 3 RD INTERNATIONAL CONFERENCE ON COMPUTATIONAL ELECTROMAGNETICS AND ITS APPLICATIONS (ICCEA), NOV. 1-4 2004, BEIJING, CHINA.....	99
PAPER PC5: V. A. CHOULIARAS, J. L. NUNEZ-YANEZ, T. R. JACOBS AND ASHWIN K. KUMARASWAMY, 'CONFIGURABLE MULTIPROCESSORS FOR HIGH-PERFORMANCE MPEG-4 VIDEO CODING', PROCEEDINGS OF THE IEEE ANNUAL SYMPOSIUM ON VLSI, MAY 11-12 2005, TAMPA, FLORIDA, USA	105
PAPER PC6: ASHWIN K. KUMARASWAMY, V. A. CHOULIARAS, T. R. JACOBS, AND J. L. NUNEZ-YANEZ, 'SYSTEM-ON-CHIP DESIGN FRAMEWORK (SDF) UNIFYING SPECIFICATION CAPTURE AND DESIGN MODELLING', PROCEEDINGS OF THE 2005 ELECTRONIC DESIGN PROCESSES (EDP) WORKSHOP, APRIL 6-8, MONTEREY BEACH HOTEL, MONTEREY, CALIFORNIA, USA.....	108
PAPER PC7: V M DWYER, S AGHA AND V. CHOULIARAS, 'LOW POWER FULL-SEARCH BLOCK MATCHING USING REDUCED BIT SAD VALUES FOR EARLY TERMINATION', PROCEEDINGS OF MIRAGE 2005 INTERNATIONAL CONFERENCE ON COMPUTER VISION/COMPUTER GRAPHICS COLLABORATION TECHNIQUES	115
PAPER PC8: TOM R. JACOBS, VASSILIOS A. CHOULIARAS AND JOSE L. NUNEZ, 'A THREAD AND DATA-PARALLEL MPEG-4 VIDEO ENCODER FOR A SYSTEM-ON-CHIP MULTIPROCESSOR', ACCEPTED FOR ORAL PRESENTATION AT THE IEEE 16TH INTERNATIONAL CONFERENCE ON APPLICATION-SPECIFIC ARCHITECTURES AND PROCESSORS (ASAP 2005), SAMOS, GREECE, JULY 23-25 2005	122

PAPER PC9: S. R. PARR, K. KOUTSOMYTI, V. A. CHOULIARAS, J.L. NUNEZ, D. J. MULVANEY, ' <i>CONFIGURABLE SCALAR AND VECTOR COPROCESSORS FOR ACCELERATING THE G.723.1 AND G.729.A SPEECH CODERS</i> ', ACCEPTED FOR ORAL PRESENTATION AT THE IASTED INTERNATIONAL CONFERENCE ON SIGNAL AND IMAGE PROCESSING (ACIT-SIP), NOVOSIBIRSK, RUSSIA, JUNE 20-24, 2005	129
GROUP RJ: UNDER-REVIEW JOURNAL PAPERS	135
APPENDIX A: COMPLETE LIST OF AUTHOR PUBLICATIONS.....	138
A.1 ACADEMIC JOURNAL PUBLICATIONS (PUBLISHED)	138
A.2 ACADEMIC JOURNAL PUBLICATIONS (UNDER REVIEW/ACCEPTED FOR PUBLICATION).....	138
A.3 REFEREED CONFERENCE PUBLICATIONS (PRESENTED/ACCEPTED)	139
A.4 PATENT APPLICATIONS	143
A.5 NON-REFEREED CONTRIBUTIONS	144

Chapter 1

Introduction to the Thesis

The term 'embedded systems' typically refers to non-visible, programmable computers that control a particular function in a higher order system. Examples of such systems are automobile Engine Management System (EMS), controllers embedded within household appliances, distributed arrays of sensors, programmable baseband processors in mobile terminals and chip-multiprocessors (CMP) in PCI-X add-on cards for scientific computing. The range and diversity of embedded systems is enormous and there are very few (if any) application domains where embedded 'intelligence' is not needed. Embedded systems are implemented in a number of technologies ranging from printed circuit board (PCB)-based systems to single-die and multi-die (stacked) Systems-on-Chip (SoCs). In this thesis, the term embedded system is considered synonymous to high-performance systems-on-chip.

Parallelism is a common characteristic of modern media and telecom applications in the embedded domain. It comes in a number of forms, expressed either by the human programmer or automatically exposed by the high-level language compiler. There are four universally accepted forms of parallelism namely, *Instruction Level Parallelism* (ILP), *Data Level Parallelism* (DLP), *Thread Level Parallelism* (TLP) and *Process Level Parallelism* (PLP). The distinction amongst these forms of parallelism relates to the fraction of the software application that can be executed in parallel (PLP and TLP), the spatial arrangement of the data items processed by the application as well as their dependencies (DLP) and finally, the register operand and execution resource requirements of the application binary (ILP). This thesis elaborates on thread level and data level parallelism as these have been found to be the most relevant to the application domain of interest, and discusses mechanisms around a non-rigid software-hardware interface for the benefit of exploiting these most important forms of parallelism.

The thesis consists of three groups of papers. The first group includes five published journal papers and the second group includes nine accepted/presented national and international conference papers. Both groups consist of contributions which belong to one of three thematic areas namely, *data level parallelism*, *thread level parallelism* and the *software-hardware*

interface. A final group includes references to other journal publications by the author and his research group, currently under review or in the final acceptance phase.

1.1 Thematic Areas

This section elaborates on the three major thematic areas covered by the published works. A reference is also made to *instruction level parallelism* but the thesis does not substantially address this as the principal benefits in theoretical performance and thus, execution time, are realized primarily via the exploitation of TLP and DLP.

1.1.1 Data Level Parallelism

This thematic area includes contributions found in papers PJ2, PJ3, PC2, PC3, PC4, PC8 and PC9. DLP extraction and exploitation is a well studied subject in the domain of scientific computing, computational fluid dynamics (CFD), financial analysis and more recently, genome modelling. It has become a potent performance leverage in feature-rich, SoC-based consumer products over the past few years as the media content processed by such appliances and transmitted over existing and emerging wired and wireless networks, has increased dramatically.

The unwillingness of industry to shift to a programming paradigm that explicitly exposes parallelism (resembling the unwillingness of the same industry to adopt a self-timed design methodology as a superior design paradigm from the power management and timing-independence perspectives) initially led to limited exploitation of DLP and, where it was explored, that was using ad-hoc, manual methodologies. This rigidity can be primarily attributed to the software engineering establishment which uses well known high-level-languages (HLLs) such as C and C++ which simply don't provide the necessary semantics. In fact, both C and C++ rely heavily on constructs such as pointers which are potentially detrimental to the automatic detection of thread and data parallelism. It is interesting to point out that the scientific community has long identified DLP as being capable of significantly improving performance and has developed FORTRAN vectorizing compilers to automatically take advantage of vector floating-point hardware. In the consumer electronics area, there are few high-level language compilers capable of exposing DLP and these come from established vendors such as *Tensilica* and dynamic start-ups such as the Philips spin-out *SiliconHive*. In

the latter case, I believe that the compiler transforms the DLP to ILP in order to schedule the very wide Avispa+ family of Ultra Long Instruction Word (ULIW) CPUs. A number of recent microprocessor start-ups acknowledge the benefit of automatically exploiting such parallelism; however they remain suspiciously cryptic as to their technology's ability to automatically uncover that parallelism.

To address these issues and exploit DLP, the Electronic Systems Design Group at Loughborough University embarked in an initially manual process of quantifying the amount of data parallelism in media and telecom workloads. Our studies characterized precisely the theoretical performance benefit of vectorized media, telecom and scientific workloads and resulted in a systematic methodology for developing vector instruction set architecture (ISA) extensions for established architectures.

The results of this work have shown benefits in media and telecom applications when the starting points are the publicly-available implementations such as the MPEG-2 TM5, the open-source MPEG4 XViD and the open-source X264 codes. Communication with industry has established that the algorithms that are suitable for SoC-based products are highly-optimized and assembly-recoded versions of such publicly available codes. An industrial colleague, Mr. John Edwards from Motorola UK, quantified the performance improvement going to hand-coded, optimized assembly versions from publicly available codes to be within an order of magnitude (factor of ten). As we don't have access to such optimized implementations, we postulate that the algorithmic benefit achieved in our studies will be achievable on the hand optimized codes as well. It is interesting to state that we realized tremendous benefits in highly regular scientific applications such as 3D transmission line modelling (TLM) codes. These are short codes (kernels) that are characterized by high spatial and temporal regularity in data accesses as well as very high floating-point computational requirements.

To harness DLP within the constraints of an SoC embedded system, we architected and currently develop a number of data-parallel accelerators for media, telecom and scientific workloads with a clear focus on tightly integrating such coprocessors to an open-source, configurable 32-bit Sparc V8 compliant RISC CPU core, developed for the European Space Agency by Gaisler Research in Sweden.

1.1.2 Thread Level Parallelism

Thread-level parallelism refers to the parallel execution of parts of the control flow graph (CFG) of the software application via a collection of processors (processor contexts) typically (but not necessarily) in a shared memory configuration, while fully observing sequential execution semantics (data dependencies). Typical SoC architectures that can exploit and benefit from this form of parallelism are chip-multiprocessors (CMP), multi-threaded processors (MT) or multithreaded chip-multiprocessors (CMP-MT). A very dramatic example of the later architecture is the 'Niagara' CPU under development by Sun Microsystems. The processor includes eight scalar Sparc V9 CPU 'cores', each being a 4-way MT-processor in a shared memory configuration. As a result, a single device accommodates 32 CPU contexts allowing for the dramatic improvement in the execution performance of threaded server codes. Though not an embedded system in its truest sense, such CMP-MT configurations are expected to become dominant in 90nm silicon nodes and beyond.

This thematic area, in the contexts of current and emerging video coding standards, is discussed in contributions PC5 and PC8. In addition, the Electronic Systems Design Group is transforming the high performance H264 standard and its X264 open-source implementation to a thread-parallel version. The potential of TLP is evident from the already published research as well as our ongoing efforts in the field. In fact, data suggest that under ideal communications (execution on a PRAM model), the thread-parallel MPEG-2 TM5 implementation provides significantly better performance than the data-parallel version.

Finally, we observe in the video coding subset of the workloads that TLP is a complementary form of parallelism to parallelism at the data level (typically available at inner loops) and can be found at the outer loop levels of the compute-intensive areas of the application. This observation suggests that a SoC chip vector multiprocessor (multi-vector computer) in a shared memory configuration is a very potent engine for the real-time execution of such codes. I have worked independently and for a number of years in the area of vector multiprocessors and the outcome is the *SS_SPARC* project which is briefly discussed in the 'future research' section.

1.1.3 *The software-hardware interface*

This thematic area, covered in PJ1, PJ2, PJ3, PJ4, PC1, PC2, PC3 and PC4 is probably the most conceptually difficult area to appreciate for established software developers who normally deal with fixed ISAs. Our research has extended these ideas by proposing vector ISA extensions, primarily designed around a private (non-architecturally-specified) register file. These data-parallel extensions are encapsulated within a generic microarchitectural framework (coprocessor) that is closely-coupled to the 32-bit Leon-2 CPU. Nowadays, these ideas are becoming more widespread and we witness the emergence of an increasing number of CPU intellectual-property (IP) start-ups that advocate ISA extensibility. In particular, *Stretch Inc.* has produced an application-specific integrated circuit (ASIC) which includes a modified Tensilica Xtensa Core and an uncommitted (embedded) field-programmable gate array (FPGA) fabric. The device is complemented by automatic tools (including a C compiler, and it appears, a vectorizer) which, given the target application, will automatically compile, profile, and generate data-parallel datapaths and scalar custom instructions in the reconfigurable fabric with the remainder of the application running on the Xtensa core. This is a level of automation previously unheard of (perhaps with the exception of the latest Tensilica C compiler and FLIX execution semantics) which can potentially provide tremendous power to system developers, without the need to go the ASIC route.

I strongly advocate the route taken by Stretch with the exception that automation should be applied at 'design time'. This would provide much improved performance, both from a microarchitecture as well as from a performance point of view, but at the expense of having to produce an ASIC. This is the route followed by Tensilica and currently looked at by *ARC International*. The Electronic Systems Design Group will focus on automatic ISA Extension development in the coming months.

1.1.4 *Instruction Level Parallelism*

Microprocessor manufacturers have strived to extract as much ILP as possible from sequential applications in programmable, pipelined architectures. ILP is a low-level form of parallelism in the sense that, unless the human programmer is willing to code an application explicitly in

assembly, ILP can't be controlled and is thus left to the capabilities of the compiler to expose it and that of the underlying ILP microarchitecture to execute it.

We have witnessed an explosion in the capability of ILP architectures in desktop and large-scale systems. Taking as an example the largest microprocessor vendor, the first major attempt to exploit ILP was with the Pentium P5 microarchitecture; a dual-issue, statically-scheduled CPU with a high performance Floating Point Unit (FPU). A number of other microprocessor vendors have produced similar microarchitectures the most striking of which was the Alpha architecture whose first implementation, the 21064, shattered the clock frequency records when it was first introduced in 1992 at an operating frequency of 200 MHz. Subsequent implementations of the X86 and Alpha architectures materialized, each increasingly more complex and using more sophisticated dynamic features for exposing and exploiting the limited ILP available in desktop workloads.

In the embedded domain however, there have been few significant attempts in developing ILP-capable architectures. This is primarily attributed to the perceived lack of ILP and the abundance of other forms of parallelism such as TLP and DLP in the target application area as well as the very limited power budgets of battery-powered embedded devices. I do however believe that within their organizational constraints and engineering budgets, embedded CPU vendors could still extract performance benefits by implementing low-order ILP pipelines.

Though not directly presented in this thesis, the author is working in a high-performance ILP, DLP and TLP-capable RISC-based ASIC processor architecture which is highly parameterized in both the architecture and microarchitecture axes.

1.2 Methodology

The methodologies followed across all three research themes originate from established industrial practice that I was exposed to when I was working as a professional engineer for ARC International and as an ASIC designer for INTRACOM. One of the major issues at that time, and an issue I am confident applies to most major embedded CPU IP providers, was the lack of precise data on the potential benefit of exploiting the forms of parallelism discussed above. Clearly, techniques such as profiling are well established, however at the time, ARC did not address the issues of vectorization, manual or automatic vector ISA design, threading

and execution on a parallel-RAM (PRAM) model. To the best of my knowledge, these issues were not being addressed at ARM, MIPS and Tensilica at that time.

After joining the Department of Electronic and Electrical Engineering at Loughborough University, I focused on developing the infrastructure that I deemed important in an effort to quantify and exploit parallelism at all levels. This has led to two major contributions in the domain of architectural simulators (ISS) for embedded CPUs. Starting with the publicly available SimpleScalar computer architecture research tools, I developed a systematic methodology for adding additional programmer-visible state to the default ISS (sim-profile). This led to the development of a new simulator known as *sim-vector* which has become the standard tool by which the Electronic Systems Design Group at Loughborough has been studying vectorized workloads. Subsequently, I re-architected the sim-profile simulator, part of the simpleScalar tools which resulted in a second-generation simulator known as *sim-system*. Sim-system can be considered as an Exclusive-Read, Exclusive-Write (EREW) Parallel RAM (PRAM) simulator and has been utilized very successfully in studying statically-threaded workloads. It is interesting to note that the threading and execution methodologies are interoperable with the shared-memory OpenMP API.

In addition, I developed most of the scripting support infrastructure to run the newly developed simulators and automatically collect the results and trained all my Ph.D. students and senior researchers to actively study DLP and TLP. The simulators and the collection of scripts are used continuously within the Electronics Systems Design Group.

1.3 Further work

This section briefly presents ongoing and near-future research initiated and managed by the author:

1.3.1 Advanced software optimization tools

The issue of *automatic software optimization*, which includes vectorization along the inner loops, threading across the outer loops, compiler-directed prefetching for hiding memory

latencies, automatic data alignment for multi-banked on-chip memory hierarchies and custom ISA extensions constitutes what many CPU architects would consider to be the *Holy Grail of Computer Architecture*. Research in the past thirty years has established solutions to some of these issues however, I strongly advocate a holistic approach, one that would be independent of ISA and application domain. Such tools, if they existed, would assist next-generation SoC designers in precisely studying, profiling, parallelizing and optimizing their codes, in a generic way, to match a generic and highly scalable, programmable, SoC platform. Subsequently, co-simulation of the generic platform and the optimized software and design space exploration would yield the final optimization parameters that would lead to a near-optimal, software-based solution.

1.3.2 The System Design Framework

The *System Design Framework* project has been designed to give the Electronic Systems Design Group a powerful tool to probe the microarchitecture space of next-generation streaming Systems-on-Chip. The SDF is the cycle-accurate back-end to sim-system and will permit the modelling of emerging (5-10 years projection) microarchitectures with near-RTL accuracy. Our studies on vector and threaded workloads were carried out with sim-system (and its predecessor, sim-vector) which gave us confidence that exploitation of parallelism at all levels is the way to approach the development of VLSI systems of substantial processing capability and complexity. We are actively pursuing this research project and currently we are in talks with BAE Systems to finalize support and commitment to our approach. The successfully blending of object-oriented system design methodologies as advocated by my colleagues Dr. David Mulvaney and Mr. Ashwin Kumaraswamy has opened the door to many exciting possibilities for the particular work. This area appears to be of great importance and this is reflected on the Gatsby Grant, awarded to the Group as pathfinder funding to further develop this technology. In addition, we are completing the patent application process for the Universal Modelling Language (UML)-to-SystemC part of the flow and we target the United States Patent and Trademarks Office (USPTO).

1.3.3 *Fault-Tolerant Parametric Multiprocessor Kernel for mission-critical applications*

This is an active project with my colleagues, Dr. James Flint and Mr. Emmanuel Touloupis. Over the past two years, we developed a unique, fault-tolerant version of the Leon-2 CPU system that shows substantial tolerance against single-event and multi-event upsets. The microarchitecture is based on the triplication of the execution pipeline which differentiates our method from other techniques which involve triplication of the microarchitecture state flip-flops. The pipeline is characterized by a distributed voting scheme for the per-cycle validation of all microarchitecture state across all three pipelines. During normal operation, the system executes the application software on all three pipelines but with only one pipeline committing architecture state (register file and memory values). On detection of a single or multi-bit error, the distributed control logic automatically re-configures the triplicated pipeline by taking out the particular datapath that developed the fault and entering a special *pair* mode of operation. This operation takes only one cycle. Subsequently, the faulty pipeline is re-introduced after a programmable number of cycles at which point the system enters again TMR mode of operation. If a further fault is detected during operation in *pair* mode, the system enters an automatic restart procedure which leads to a software reset and the rebooting of the CPU.

We have successfully developed the architecture, implemented the microarchitecture, produced ASIC macros of the CPU and collected a significant number of simulation results when executing the automotive subset of benchmarks in the MiBench benchmark suite. Our findings have been reported on national and international conferences [7 15 17] and are the subject of an ongoing patent application. In addition, our industrial collaborators, MIRA Ltd, have expressed interest to develop this idea further into automotive X-by-wire and they currently champion the patenting process. Dr. Flint and Mr. Touloupis have agreed to pursue this further through the development of a configurable, SoC execution kernel and associated infrastructure for such systems. We are drafting an EPSRC proposal for this system and a US Patent for the existing multi-pipeline, single-context, fault-tolerant CPU.

1.3.4 The Ultimate ASIC CPU approach: The SS_SPARC

This is a major effort to develop a very high performance configurable, extensible CPU system that efficiently handles all the forms of parallelism as discussed in this chapter. I have been working on this over a number of years and the current state of the design is satisfactory. The CPU is a multi (five)-issue, in-order-dispatch, out-of-order commit, Sparc V8 compliant microarchitecture parameterized as to the number of CPU contexts, memory hierarchies, and DLP infrastructure. I envision this design as a replacement for the arbitrary choice of microcontroller/CPU/DSP combinations in current and next generation SoCs due to its unique parameterization, streaming and execution bandwidth, and configurability and extensibility options. The licensing model of the design is yet to be finalized and the route by which it will be made available is also not clear at present.

1.4 Author Contributions in the published works

This section discusses in detail my contribution to each of the published works.

Paper PJ1

The process of developing two scalar ISA extensions to complement the SimpleScalar and Sparc V8 ISAs along with the embodiment of the first ISA in the newly-developed sim-vector ISS was my responsibility. Dr. J. Nunez provided assistance in the development of the private register file solution and its subsequent benchmarking. In addition, I developed the scripting infrastructure to automatically run the workloads over all configurations and established the methodology by which research in ISA design is carried out in the Electronic Systems Design Group.

Paper PJ2

The engineering methods of developing a custom vector ISA and subsequently, a vector accelerator for the ITU-T G.729.A and G.723.1 speech coding algorithms were my primary contributions in this work. The whole concept originated after long conversations in 2002

with one of the senior software experts at ARC International's DSP group, Dr. Dariush Baghbadrani, during which we sought ways of addressing the limitations of the dual 16-bit ARC Tangent A4 CPU DSP engine. The methodology followed since then led to the development and continuous enrichment of 'sim-vector', the group's proprietary ISS. In addition, I developed the scripting infrastructure to automatically run the workloads over all configurations and collect the simulation results.

Paper PJ3

My contributions in this work related to the development of the vector ISA for the acceleration of MPEG-2, MPEG-4 (XViD) and H264 (proprietary ST Microelectronics implementation), the update of sim-vector and subsequent collection of results, the development of the common vector coprocessor for accelerating the inner loop of motion estimation (ME) in all algorithms, the development of the processor-coprocessor interface, RTL coding, front-end synthesis and back-end implementation. Dr. Nunez provided significant help in the vectorization process of the MPEG-4 (XViD) workload.

Paper PJ4

This contribution was in a slightly different area to my direct research interests. As a result, my contributions in this work were primarily around the processor-coprocessor interface, the precise exception mechanism and ways to introduce both programmer (visible) and microarchitecture (invisible) state in a side-pipeline, running in parallel to the main CPU. In addition, I performed the ASIC synthesis front-end tasks and all associated back-end operations including *physical synthesis* and detailed routing.

Paper PJ5

My contributions to this work related to input to the concept of different scanning-orders and methods of minimizing the (expected) very high cache misses ratios and associated latencies, the setting up of the simulation infrastructure based around sim-vector and the automatic

collection of results over a period of time. My results were subsequently compared to those collected by the other two researchers which used both abstract performance metrics as well as real-time measurements and correlated reasonably well.

Paper PC1

This paper preceded contribution PJ1 and discusses the development of one scalar accelerator for the G.729.A speech coding standard. My contributions were in the development of that scalar ISA, its introduction in the sim-vector ISS and the definition of the microarchitecture and means of introducing a side-datapath to the default Leon-2 CPU. I also developed the scripting infrastructure to automatically run the workloads over all input vectors and collect the results

Paper PC2

Algorithmic profiling, vector ISA definition, parameterization, microarchitecture specification and implementation were my responsibility in this work. Mr. Sharukh Agha developed 14 sub-sampling (fast) ME algorithms which were subsequently studied, in the context of a parametric vector architecture. These results were not published in this conference but are under scrutiny by Dr. Vince Dwyer, Mr. Agha and the author and scheduled to be submitted to the IEEE 2006 International Conference on Consumer Electronics (ICCE 2006).

Paper PC3

In this paper, I contributed to the algorithmic profiling, vector ISA definition, the parameterization, microarchitecture specification and ASIC implementation of the combined 32-bit RISC CPU and the parametric vector accelerator.

Paper PC4

My contribution in this work was in the specification of the way the code should be (explicitly) vectorized, the vector floating point ISA, microarchitecture (floating-point datapath), the combined pipeline and its communication mechanism with the scalar 32-bit RISC CPU. Dr. James Flint provided the initial 3D TLM kernel which was jointly re-written to expose the DLP (in the inner loops) and the TLP (in the outer loops).

Paper PC5

The design and implementation of both the multi-threaded instruction set simulator (MT-ISS), the barrier mechanism and the synchronization principle were my contributions in this work. In particular, the simplescalar toolset was re-engineered to include a (configurable) number of extra CPU contexts, additional state was added to each context to facilitate synchronization (hardware-like barrier mechanism) as well as a number of machine states (sleep modes) were introduced. Sim-system proved to be an invaluable tool in all out studies in threaded consumer and scientific codes and has eventually become our mainstream simulator. A further, ongoing effort involves the production of a single static and multiple dynamic execution traces from sim-system which are consumed by a cycle-accurate back-end. This effort will allow us to model arbitrary System-on-Chip multiprocessors (CMP), multithreaded processors (MT) or multithreaded multiprocessors (CMP-MT) as well as arbitrary memory hierarchies and interconnect.

Paper PC6

My contribution in this work was twofold: Firstly, in the refinement and ‘industrialization’ of the core tool developed by Mr. Ashwin Kumaraswamy, I proposed a different ESL target (SystemC) after extensive experimentation with an industrial-strength SystemC compiler/synthesizer. As a result, the flow was streamlined and the loop was closed, from system level specification capture in UML, all the way down to GDS. The second contribution was in the design and implementation of the multithreaded instruction set simulator used to collect the theoretical results from the MPEG-4 video encoder. The

combined effort of specification capture and SoC modelling has lead the Electronic Systems Design Group to proposed a holistic approach to the grant challenge of specification capture, all the way to silicon, of highly complex, next generation SoC platforms for media and scientific workloads.

Paper PC7

My contribution was in the deployment of custom vector ISA extensions for the efficient execution of a number of algorithmic (sub-sampling) ME algorithms developed by Mr. Sharukh Agha. The theoretical study on the RBSAD was performed by Dr. Vince Dwyer. In particular, Mr. Agha developed a substantial number of sub-sampling ME algorithms and variations which were profiled and evaluated for the vector ISA extensions identified in PC2. A larger number of results are available which will be published at a future IEEE Consumer Electronics Conference.

Paper PC8

Algorithmic profiling, scripting support, implementation of the PRAM simulator and overall guidance were my responsibilities in this work. Mr. Tom Jacobs very meticulously parallelized the convoluted video coding workload and Dr. Nunez worked on the vectorization aspect of the coder.

Paper PC9

My contribution in this work related to the setup of the scripting infrastructure, the development of the simulators, the profiling and evaluation processes and the overall guidance and supervision of the two researchers, Mr. Simon R. Parr and Mrs. K. Koutsomyti.

Group PJ: Published Journal Papers on Parallelism and the Software/Hardware Interface in embedded systems

PJ1:

V. A. Chouliaras, J. L. Nunez, *'Scalar Coprocessors for accelerating the G723.1 and G729A Speech Coders'*, IEEE Transactions on Consumer Electronics, Vol. 49, Issue 3, Aug. 2003, pg. 703-710, ISSN:0098-3063

This paper discusses a number of scalar ISA extensions developed to accelerate the ITU-T G.723.1 and G.729.A speech coding standards. In particular, results are presented for two scalar coprocessors, one with and one without a private, scalar register file, which are tightly attached to a 32-bit RISC CPU. The results demonstrate that the coprocessor employing a private register file achieves superior performance due to relieving the pressure on register allocation during the compilation process. Both coprocessors are designed to be tightly attached to the open-source Leon-2 Sparc V8 compliant CPU. A custom coprocessor channel is presented and the microarchitecture is detailed.

PJ2:

V. A. Chouliaras, J. L. Nunez, K. Koutsomyti, S. R. Parr, D. J. Mulvaney, S. Datta, *'On the development of a custom vector accelerator for high-performance speech coding'*, IEE Electronic Letters, Vol. 40, Issue 24, 25 Nov. 2004, pg 1559-1561

This is the first journal contribution in which results are presented for a collection of vector extensions to the Sparc V8 ISA for accelerating the G.723.1 and G.729.A speech coding standards. Results indicate that a parametric data-parallel architecture and microarchitecture, 32 bytes wide, is sufficient to capture the greatest amount of DLP in the speech coding workloads. The results in this work are complementary to the data presented in PJ1.

PJ3:

V. A. Chouliaras J. L. Nunez, D. J. Mulvaney, F. Rovati, D. Alfonso, *'A Multi-standard Video coding accelerator based on a vector architecture'*, IEEE Transactions on Consumer Electronics, Vol. 51, Issue 1, Feb 2005, pg 160-167, ISSN:0098-3063

This work quantifies the DLP in a significant subset of embedded workloads, namely media (consumer) applications and in particular, transform-based video coding. The paper discusses the profiling of the MPEG-2 TM5, MPEG-4 (XViD) video coders as

well as a proprietary implementation of the H264 video encoder, supplied by ST Microelectronics. Subsequently, a set of approximately 45 vector instruction extensions to the Sparc V8 ISA are identified and developed and a pipelined microarchitecture is proposed to implement them. The paper includes a VLSI implementation of the combined processor-coprocessor design, implementing a subset of the MPEG-4 vector ISA, targeted at a high-performance 0.13 μm CMOS process

PJ4:

J. L. Nunez, V. A. Chouliaras, '*High Performance Arithmetic Coding VLSI Macro for the H264 Video Compression Standard*', IEEE Transactions on Consumer Electronics Vol. 51, Issue 1, Feb 2005, pg 144-151 ISSN:0098-3063

An interesting alternative to accelerating the data-parallel parts of the H264 video coding standards is presented in this work. In particular, the complexity, in terms of dynamic instruction count as well as function calls, of the arithmetic coding process is measured in the H264 video coding standard JM 9.2 reference implementation. We subsequently developed a new, hardware-focused arithmetic coding algorithm realized as a multiplication free, non-stalling pipeline, able to process one bit per cycle while maintaining the original arithmetic coding efficiency of the H264 reference implementation. The functionality of the arithmetic coding engine is encapsulated in a custom coprocessor which attaches to our default RISC CPU. Issues on maintaining a precise exception model for the software are identified and solved at the microarchitecture level.

PJ5:

Grecos, C., Saparon, A. and V. A. Chouliaras., '*Three novel low complexity scanning orders for MPEG-2 full search motion estimation*', Real Time Imaging, 10, February 2004, pp 53-65, ISBN 1077-2014

This paper presents an interesting alternative, based purely on algorithmic optimizations, to the computational complexity issues apparent in real-time video encoding. In particular, three scanning orders of similar complexity are identified for the ME process in MPEG-2 TM5 and it is shown that they reduce by approximately 7.1% the number of examined macroblocks, potentially reducing data cache misses.

Paper PJ1: **V. A. Chouliaras**, J. L. Nunez, '*Scalar Coprocessors for accelerating the G723.1 and G729A Speech Coders*', IEEE Transactions on Consumer Electronics, Vol. 49, Issue 3, Aug. 2003, pg. 703-710

Scalar Coprocessors for Accelerating the G723.1 and G729A Speech Coders

Vassilios A. Chouliaras and Jose Nunez, Member, IEEE

Abstract — We investigate two scalar coprocessors for accelerating the ITU-T G723.1 and G729A speech coders. Architecture space exploration indicates up to 72% reduction in the total number of instructions executed through the introduction of custom instructions and small changes to the C reference code. The accelerators are designed to be attached to a configurable embedded RISC CPU where they make use of the host register file and Load/Store Infrastructure¹.

Index Terms — Coprocessor, Embedded systems, RISC CPU, Speech coding.

I. INTRODUCTION

Speech compression is utilized in a multitude of applications including amongst others VoIP networks and digital satellite systems. Typical consumer products comprise multimedia terminals, digital dictation machines, videophones and IP phones. The G723.1 recommendation [1] in particular was designed to standardize telephony and videoconferencing over public telephone lines (POTS) and is part of the ITU H.324 standard.

This work investigates the benefit, in terms of complexity reduction, of architecture (instruction) extensions for the efficient execution of the above vocoders, building on previous work by the authors [6]. The identified extensions are implemented as coprocessors, tightly-coupled to a configurable, embedded RISC processor.

There is a significant body of research into application acceleration via targeted coprocessors: application domains are diverse, ranging from cryptography [12], maze-routing [7] to high-end video processing [19]. Previous research into the efficient execution of speech coders include [13] and [14] which describe the necessary changes in the ITU reference code when targeting very high-performance, off-the-self digital signal processors. [15] describes a semi-automated chip-synthesis flow targeting a horizontally microprogrammed (VLIW) embedded DSP architecture, capable of executing one multiply-accumulate operation per clock cycle. The workload in this case was the GSM half-rate speech coder.

Our research is a continuation of [6] which describes instruction set extensions, implemented in a moderate-complexity datapath (coprocessor) attached to a configurable embedded processor. We have investigated a second coprocessor configuration which includes a private register file. Results indicate that the new configuration is superior the previously reported method.

V. A. Chouliaras is with the Department of Electronic and Electrical Engineering, University of Loughborough, Loughborough, Leicestershire LE11 3TU, UK (e-mail: V.A.Chouliaras@lboro.ac.uk).

Jose Nunez is with the Department of Electronic and Electrical Engineering, University of Loughborough, Loughborough, Leicestershire LE11 3TU, UK (e-mail: J.L.Nunez-yanez@lboro.ac.uk).

II. LPAS- BASED SPEECH CODERS

The G723.1 and G729A standards belong to the category of Linear-Prediction Analysis-by-Synthesis (LPAS) [21] speech coders. They produce low bit-rate, high-quality speech using a combination of analysis-by-synthesis techniques where the encoder (analysis) includes the decoder (synthesis) to determine the initial excitation signal, and linear prediction techniques to determine the coefficients of the speech synthesis filter. The G723.1 standard specifies a dual rate speech coder that can operate at 5.3 or 6.3 Kbps while the G729A operates at a rate fixed at 8 Kbps. Quality improves with higher bit rates although the overall performance of G723.1 at 6.3 Kb/s and G729A is similar. A clear difference in these coders is their algorithmic delay where the total one-way delay of G729A of 25 ms compares favorably with the 67.5 ms of G.723.1. Technically, G723.1 at 6.3 Kbps differs from G729A and G723.1 at 5.3 Kbps in the excitation model for the synthesis filter. G.723.1 at 5.3 Kbps uses multi-pulse excitation with a maximum likelihood quantizer (MP-MLQ) while G723.1 at 6.3 kbps and G729A use code excited linear prediction (CELP) [21]. CELP coders are based in a codebook that stores possible excitation sequences for the synthesis filter. This is the most common realization of the LPAS paradigm and its dataflow is depicted in figure 1.

In the figure, the original input speech is used to perform linear prediction analysis and calculate the coefficients of a tenth-order synthesis filter. The filter order models the number of resonant frequencies or formants of the transfer function of the human vocal tract. The excitation signal to the synthesis filter is obtained from two codebooks that model the initial stages of the human sound production system. An adaptive codebook is used to model the pitch structure of voice sounds originating in the vibrating vocal chords and a fixed codebook is used to model unvoiced sounds such as nasal or plosive sounds. The residual error between the reconstructed speech produced by the synthesis filter and the original input speech is then further processed by a perceptual weighting filter. The output signal from this process is then matched against the adaptive codebook elements to determine the codebook index and gain that best approximate the residual signal. The adaptive codebook contribution is removed from the residual and the same process is repeated using the fixed codebook. The index and gains for both codebooks are assembled together with the synthesis filter coefficients in the bitstream transmitted to the decoder. This processing is done for every frame of 10 ms of voice signal. The G729A decoder dataflow is illustrated in figure 2. The received bitstream is disassembled to obtain the filter coefficients and the codebook parameters. The excitation is constructed by adding the adaptive and fixed codebook vectors scaled by their gains. The excitation is then filtered through the same synthesis filter as

during encoding. Additional post-processing of the speech signal is performed to enhance its quality.

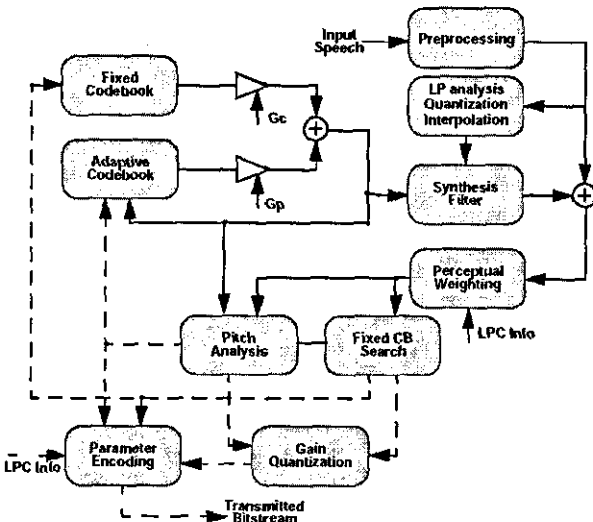


Figure 1: G729A CELP Encoder

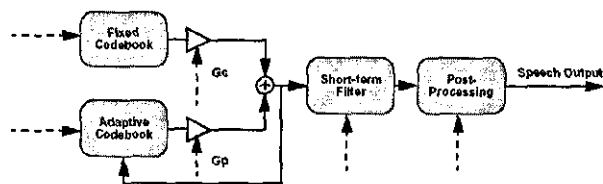


Figure 2: G729A CELP Decoder

III. PROBLEM FORMULATION

This research identifies architecture and microarchitecture requirements for the efficient implementation of the G729A and G723.1 speech coders on high-performance, low-cost, configurable microprocessors. The workloads were initially executed and profiled in native mode (Linux X86): Table 1 shows the relative amount of time spent outside the DSP emulation instructions. In order to investigate the potential acceleration of the algorithms when executing on an embedded microprocessor, the workload was recompiled for the SimpleScalar instruction set architecture (ISA) [15]. Table 2 illustrates the simulated processor profiling results. As expected, the workloads spend a significant amount of time/instructions executing the DSP emulation functions. It is clear that efficient implementation of the DSP emulation instructions on a configurable extensible microprocessor can lead to a very high-performance, targeted-architecture for the particular workloads. The small form-factor and reduced power consumption of the proposed solution makes it a very attractive candidate for replication and integration in an SoC ASIC.

Table 1: Relative amount of time spent outside the DSP emulation instructions

Algorithm	Relative time (% , native)
G723 Coder	31.3
G723 Decoder	22.8
G729 Coder	30.4
G729 Decoder	26.9

Table 2: Relative number of total instructions executed outside the DSP emulation instructions

Algorithm	Relative instructions (% , simulated)
G723 Coder	34.5
G723 Decoder	33.3
G729 Coder	34.2
G729 Decoder	37.2

This is the approach taken in this work: the Instruction Set Architecture was chosen to be precisely the DSP emulation instructions as they appear in the reference source. It is summarized in table 3:

Table 3: Coprocessor ISA

Move ops	Description
Mvrc	Move RISC CPU register to coprocessor register
Mvcr	Move Coprocessor register to RISC CPU register
Mvrsv	Move RISC CPU register LSB to coprocessor overflow
Mvcvr	Move coprocessor overflow to RISC CPU register LSB
Data ops	Description
Sature	32-16 bit ITU saturate
Add	16-bit add and saturate
Sub	16-bit sub and saturate
Abs_s	16-bit absolute value
L_abs	32-bit absolute value
Shl	16-bit Shift-left with negative shift support and saturation
Shr	16-bit shift-right with negative shift support and saturation
Negate	16-bit negation
Norm_s	16-bit normalization calculation
Norm_l	32-bit normalization calculation
L_add	32-bit add with overflow saturation
L_sub	32-bit sub with overflow and saturation
Mult	16x16->16 signed multiplication with overflow and saturation
L_mult	16x16->32 signed multiplication with overflow and saturation
L_mac	16x16->32 multiplication and 32-bit summation with overflow and saturation
L_msu	16x16->32 multiplication and 32-bit subtraction with overflow and saturation
Miscellaneous ops	Description
Clv	Clear sticky overflow bit
Setv	Set sticky overflow bit

IV. MICROARCHITECTURE

We have investigated two microarchitectures: One that uses the main CPU register file and another that utilizes its own. Both microarchitectures make use of the RISC memory subsystem (L1 Data cache) and are designed to be attached to a Sparc-V8 compliant SoC subsystem distributed under LGPL [10]. We choose to connect the coprocessors to the integer unit pipeline directly instead of designing them as AHB-compliant masters [11] for performance reasons: Stand-alone AHB coprocessors are very effective when working on medium to large blocks of streaming data. Although the workloads perform a lot of work on blocks of data (samples), there were many more instances where we had to insert custom assembly code into irregular (non-iterative) blocks. As a result, we opted for a very tightly-coupled configuration which accommodates efficiently both cases. High-level views of both microarchitectures are depicted in figures 4 and 6 respectively. This section discusses a number of design parameters:

A. Coprocessor Interface

The open-source embedded RISC processor lacked detailed microarchitecture documentation. Initial experimentation with the already existing coprocessor interface was inconclusive as to its ability to operate in a pipelined fashion. That would have had a detrimental effect on the performance of the coprocessors and it was therefore decided to implement a new, pipelined coprocessor interface. The newly developed coprocessor port can handle two coprocessors and is able to deliver an instruction on every cycle. External coprocessors provide flow control to the main processor through a dedicated stall signal.

The diagram of figure 3 shows a coprocessor data operation on cycle 1 followed by a host-to-coprocessor register transfer on cycle 2. In cycle 3, a coprocessor register is requested by the RISC processor but due to internal stall conditions, data are made available one cycle later than the expected time (cycle 5 instead of cycle 4). During that time, the main processor is held with the holdn signal. Finally, a second read operation, this time directed to Coprocessor 1, is initiated in cycle 6. Results are made available to the main pipeline in cycle 7.

B. Microarchitecture 1: Using the main RISC CPU Register File

This is the simplest microarchitecture since it makes use of the main RISC processor register file. This type of approach has been adopted by configurable microprocessor vendors [18] [22] and it is effectively a side-datapath with associated control, attached to the main CPU as depicted in Figure 4:

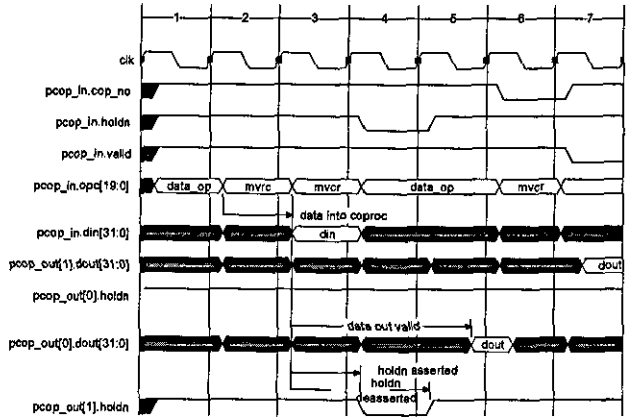


Figure 3: Pipelined coprocessor I/F

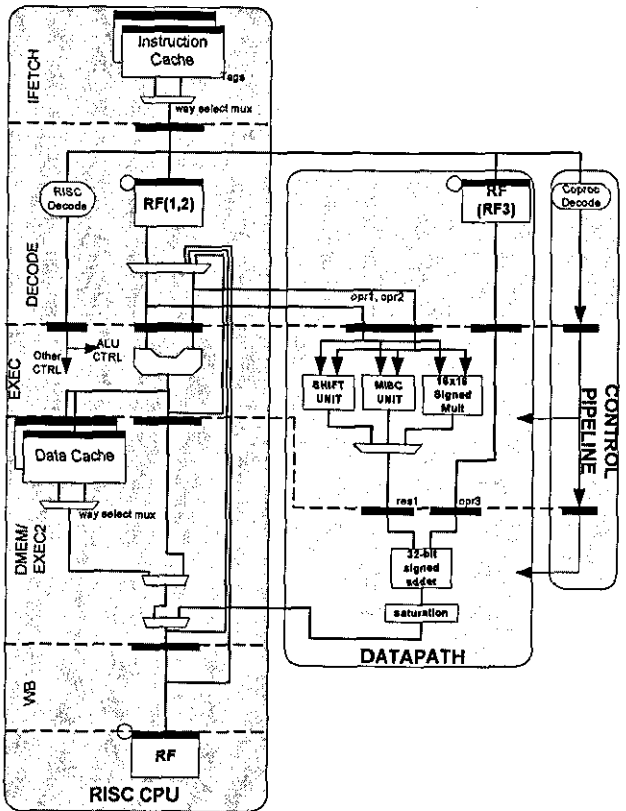


Figure 4: Microarchitecture without register file

In this case, the coprocessor consists of the Datapath and the Control Pipeline. Starting at the IFETCH stage, the main RISC processor fetches one instruction word from a multi-way set-associative instruction cache and clocks it into the instruction register. RISC and coprocessor decoding take place concurrently at the DECODE stage with the main RISC register file accessed at the falling edge of the clock. Due to the significant number of Multiply-add operations in the workload, a third read port was added to the main CPU register file to accommodate single-

cycle addition (RF3). This port is depicted as an embedded SRAM block, instantiated in the coprocessor hierarchy, clocked at the falling edge of the DECODE stage. Finally, all result bypassing takes place in this stage.

The EXEC stage is the main processing stage for both the RISC processor and the coprocessor. During this stage all non-arithmetic operations are computed in the coprocessor. In addition, the 16-bit signed-multiplication is performed. All transfers between the main RISC pipeline and the internal coprocessor state take place in this stage.

Coprocessor results are pipelined in the EXEC2 stage where the add part of the Multiply-add operation is performed along with saturation. During this stage, the L1 data cache is accessed and one 32-bit word is returned to the main RISC pipeline from the load path as depicted in the diagram. It is this stage that qualifies state updates in the coprocessor side since all possible exception conditions have been resolved. Finally, results are clocked into a staging register prior to committing to the RISC register file, on the falling edge of the clock.

C. Microarchitecture 2: Using private Register File

This microarchitecture is considerably different to the previous one due to utilizing a separate, 16x32-bit register file in addition to a more elaborate control mechanism. The coprocessor state is fully accessible from the RISC CPU and is shown in figure 5:

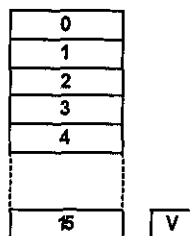


Figure 5: Coprocessor Programmers Model

It consists of sixteen 32-bit registers and a sticky overflow bit. Bi-directional transfer instructions, between the host RISC processor and the coprocessor, were added to accommodate the lack of Move-to-coprocessor/Move-from-coprocessor instructions in the Sparc V8 architecture [17].

The high-level schematic of the coprocessor with its own register file is depicted in figure 6. In this case, the coprocessor pipeline is segmented in three major sections: Front-end, Control pipeline and Datapath.

Starting from the top, the main CPU reads an instruction from the multi-way set-associative instruction cache and clocks it into the instruction register. The latched command is then decoded, both at the RISC processor and the coprocessor front-end, and register-file read-addresses are extracted. In parallel, the coprocessor decoding logic computes a number of control fields that are sent to the control pipeline.

During the EXEC/READ stage, the register file is accessed followed by operand bypassing. The resolved operands opr1, opr2 and opr3 are clocked into the operand registers where they are utilized during the first execution stage (EXEC1).

In DMEM/EXEC1, all shifting, normalization and miscellaneous operations are performed. In addition, the signed-multiplier is accessed if the command specifies that. Results are passed to EXEC2 for the second stage of execution where all arithmetic and saturation takes place.

The configuration of figure 6 permits the pipelined execution of all the commands with a latency of 1 cycle. The only exceptions are the multiply-add and multiply-subtract with saturation, which span both execution stages and have a latency of 2 cycles.

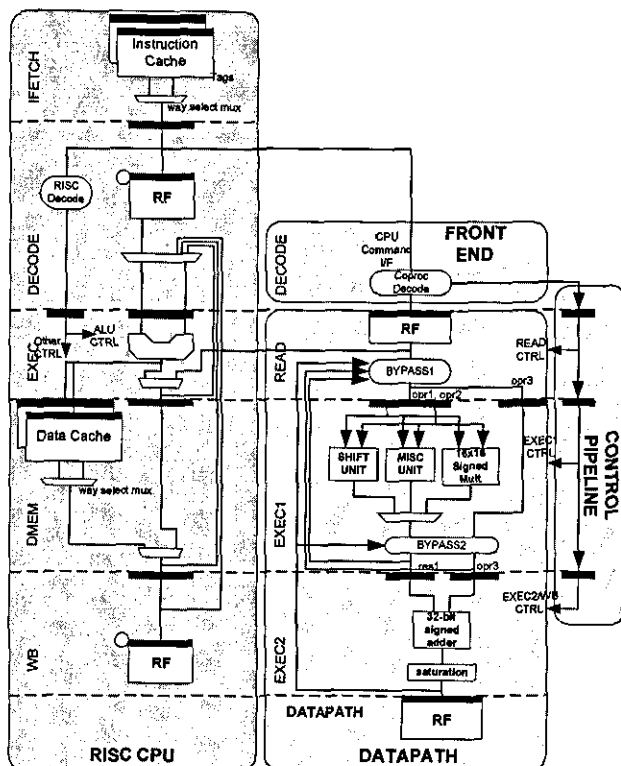


Figure 6: high-level microarchitecture

The following sections discuss in more detail the microarchitecture blocks common to both coprocessors. These include the EXEC1 and EXEC2 stages and lower hierarchical blocks.

1) EXEC1 Stage

EXEC1 includes datapath logic to perform 16x16 bit signed multiplication, all ITU shift operations and a miscellaneous block responsible for handling all opcodes not falling in the previous category. These are depicted in figure 7

a) Multiplier

This is the signed, 16-bit multiplier. Due to the highly configurable nature of the RISC processor and the portability requirements of this work, HDL constants are used to select whether the multiplier is inferred in the RTL code or instantiated. In the later case, a Booth-Encoded, Wallace-tree multiplier [20] is utilized due to the higher pipelined performance when compared to the implementations chosen by the synthesis tools.

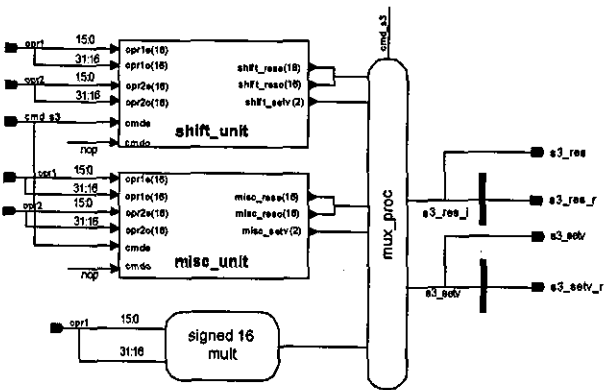


Figure 7: EXEC1 Stage

Table 4: Multiplier performance vs. architecture (MHz)

Multiplier	Unpipelined	2-stage
Synthesis/CS	204	330
Synthesis/NBW	376	
Synthesis/WALL	385	502
WALL/No		
BOOTH	345	476
WALL/BOOTH	370	574

Table 4 depicts the unpipelined and two-stage pipelined maximum operating frequency of the 16x16 signed multiplier in a high-performance 0.13 process. Our timing budget allows for the use of a non-pipelined multiplier thus, simplifying coprocessor pipeline design.

b) Shift Unit

The shift unit implements the 16 and 32-bit ITU shift operations. A particular characteristic of these operations is the ability to specify negative shift amounts resulting in a positive shift in the opposite direction. The high-level schematic of the shift unit is depicted in figure 8.

2) EXEC2 Stage

This stage performs the Add-part of the MAC instruction as well as all arithmetic and saturation. Results commit to the private register file at the end of this cycle or return to the host pipeline during stage DMEM. The common EXEC2 high-level schematic is shown in figure 9.

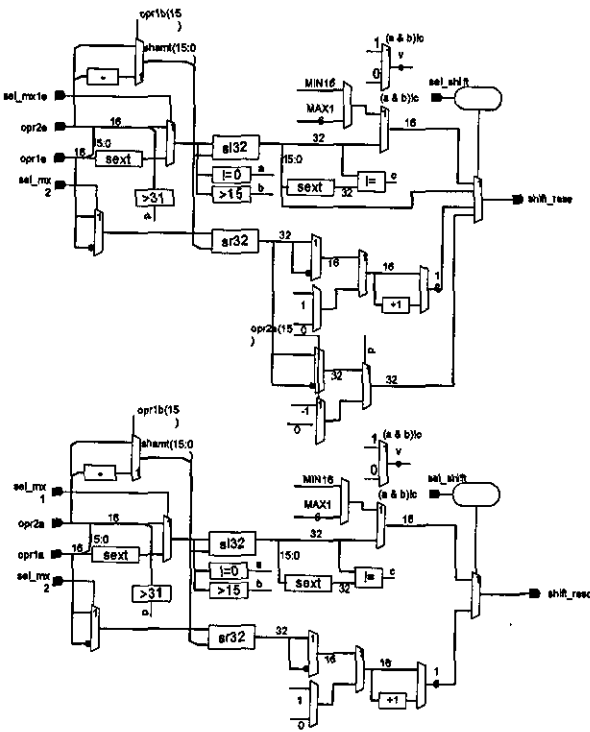


Figure 8: ITU Shifter Schematic

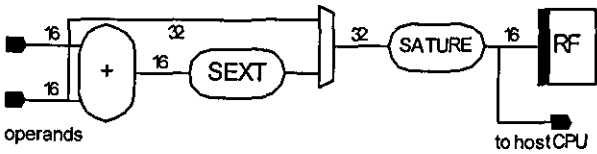


Figure 9: EXEC2 Stage high-level schematic

V. RESULTS

Results were obtained for both coprocessors at the architectural level with the baseline architecture being the Simplescalar ISA. The workloads were compiled and all ITU test vectors were validated on the standard architecture simulator (sim-profile). Tables 5 and 6 depict the number of simulated processor instructions required for each workload, for the G723.1 and G729A algorithms respectively

Table 5: G723.1 unmodified instruction count

Test vector	Instructions
Dtx53mix (mix rate)	1,063,099,834
Dtx53mix (5.3 Kbits/s)	926,595,183
Dtx63 (6.3 Kbits/s)	10,159,707,298

Table 6: G729A unmodified instruction count

Test vector	Instructions
Alghthm	62,620,904
Fixed	213,968,970
Lsp	3,977,189,411
Pitch	3,253,182,556
Tame	230,922,927

The workloads were then modified to include custom assembly instructions and a new architecture-level simulator (sim-coproc), based on the existing profiling simulator, was designed. The test vectors were again simulated and the algorithmic complexity was measured and compared to that obtained in the previous run. Fully compliance to the ITU-T test vectors was maintained at any instance.

A. Coprocessor without register file results

Tables 7 and 8 depict the average (over all test vectors), relative algorithmic complexity for both the coder and decoder of the G729A and G723.1 standards respectively when compiled and simulated for a coprocessor using the RISC processor register file.

Table 7: G729A Coder Results (average)

Normalized Complexity	Coder	Decoder	Coder Delta	Decoder Delta
SATURE	0.940	0.972	0.060	0.028
ADD	0.937	0.969	0.003	0.002
SUB	0.927	0.967	0.010	0.002
ABS_S	0.927	0.967	0.000	0.000
SHL	0.924	0.962	0.003	0.005
SHR	0.923	0.956	0.002	0.006
L_SHL	0.899	0.898	0.024	0.059
L_SHR	0.896	0.895	0.002	0.002
NEGATE	0.896	0.895	0.000	0.000
L_ADD	0.814	0.837	0.082	0.059
L_SUB	0.802	0.812	0.012	0.025
ROUND	0.796	0.801	0.006	0.011
L_ABS	0.796	0.801	0.000	0.000
NORM_S	0.796	0.801	0.000	0.000
NORM_L	0.795	0.799	0.001	0.002
DIV_S	0.792	0.797	0.003	0.002
MULT	0.771	0.784	0.021	0.012
L_MULT	0.660	0.674	0.111	0.110
L_MAC	0.534	0.580	0.126	0.094
L_MSU	0.510	0.529	0.024	0.051

Table 8: G723.1 Coder Results (average)

Normalized Complexity	Coder	Decoder	Coder Delta	Decoder Delta
SATURE	0.987	0.985	0.013	0.015
ADD	0.985	0.981	0.002	0.004
SUB	0.985	0.980	0.000	0.000

ABS_S	0.984	0.977	0.001	0.003
SHL	0.981	0.965	0.003	0.012
SHR	0.981	0.959	0.000	0.006
L_SHL	0.936	0.908	0.044	0.051
L_SHR	0.912	0.901	0.024	0.006
NEGATE	0.912	0.901	0.000	0.000
L_ADD	0.824	0.819	0.088	0.082
L_SUB	0.814	0.804	0.010	0.015
ROUND	0.809	0.788	0.005	0.016
L_ABS	0.809	0.788	0.000	0.000
NORM_S	0.809	0.788	0.000	0.000
NORM_L	0.808	0.787	0.001	0.001
DIV_S	0.807	0.787	0.000	0.001
MULT	0.806	0.786	0.001	0.001
L_MULT	0.678	0.670	0.129	0.116
L_MAC	0.563	0.541	0.114	0.129
L_MSU	0.543	0.510	0.020	0.031

The tables illustrate the fractional complexity reduction as extension instructions are added, one by one, for both coder and decoder. In the case of the G729A coder, an average architectural improvement in algorithmic complexity of the order of 49% (coder) to 47.1% (decoder) is achieved. The G723.1 standard achieves similar figures with 45.7% and 49% complexity reduction for the coder and the decoder respectively. These improvement figures do not take into account cycle-effects such as cache misses, prefetching or the possibility of multi-issue.

B. Coprocessor with private register file results

Tables 9 and 10 show the average (over all test-vectors), relative algorithmic complexity of the G723.1 and G729A coders respectively for a coprocessor with a private register file and utilizing all the defined instructions of table 3 (except division). Further substantial gains are observed: The G723.1 coder demonstrates an average relative complexity of 65% compared to the unmodified standard and an improvement of 35.6% over to the previous architecture whereas the G729A standard achieves 69% of unmodified complexity and improvement of 39.3% compared to the previous architecture. It is clear that the introduction of the coprocessor register file provided significant benefit due to reducing the register pressure compared to the previous method. In addition, a significant number of Load/Store operations were eliminated since transient values are now cached in the dedicated register file.

Table 9: G723.1 Results

Benchmark	Instruction Count (Coprocessor)	Fractional complexity
Dtx53mix (mix rate)	380,717,669	0.36
Dtx53mix (5.3 Kbits/s)	257,744,402	0.28
Dtx63 (6.3 Kbits/s)	4,261,239,585	0.42
Average		0.35

Table 10: G729A Results

Benchmark	Instruction Count (Coprocessor)	Fractional complexity
Alghm	19,765,353	0.31
Fixed	67,662,019	0.31
Lsp	1,257,199,028	0.31
Pitch	1,030,256,280	0.31
Tame	73,056,645	0.31
Average		0.31

VI. SOC SUBSYSTEM

Architecture research demonstrated the superiority of the coprocessor with a private register file. This microarchitecture is currently being implemented in RTL VHDL as a tightly-coupled coprocessor for the Leon Sparc-V8 CPU. Detailed microarchitecture analysis followed by trial synthesis confirmed that all instructions can fit in a single high-frequency cycle resulting in a latency of 1 and an initiation rate of 1. Exceptions to this are the Multiply-add/subtract instructions and the short divide with latency/initiation rate of 2/1 and 17/17 respectively. In particular, it was decided that due to the very low improvement, the iterative divider block would not be utilized. The CPU/Coprocessor attaches to a 32-bit AHB system which connects to an external host via an AHB-PCI Bridge. This is depicted in figure 10.

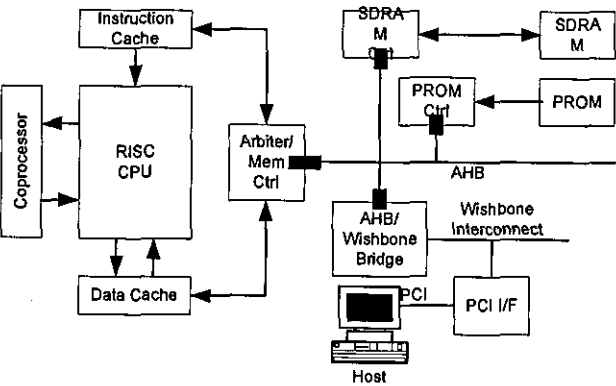


Figure 10: SoC Subsystem

The optimized speech coder and the frames to be processed are transferred with DMA from the host PC to the SDRAM memory of the RISC/Coprocessor FPGA board. After that, the RISC CPU/coprocessor combination processes the frames and stores the compressed frames in local memory (SDRAM). The compressed frames are transferred back to the PC memory for comparison with the ITU-T test vectors.

VII. SYSTEM VERIFICATION

Significant effort is spent in validating the system both at block as well as system level [16]:

A. Block-level verification

The reference code DSP emulation instructions were instrumented to produce human-readable files of their input operands, the state of the global Overflow flag and output results. These vectors were subsequently fed into the individual datapath blocks and their functionality validated on a per-workload basis.

B. System level verification

In parallel to block-level verification, system verification involved the design of a DMA controller, to transfer the embedded processor binary and frames from the host memory into the FPGA board SDRAM. The RISC processor, without the coprocessor, executed the workload and agreement with the ITU-T test vectors was obtained.

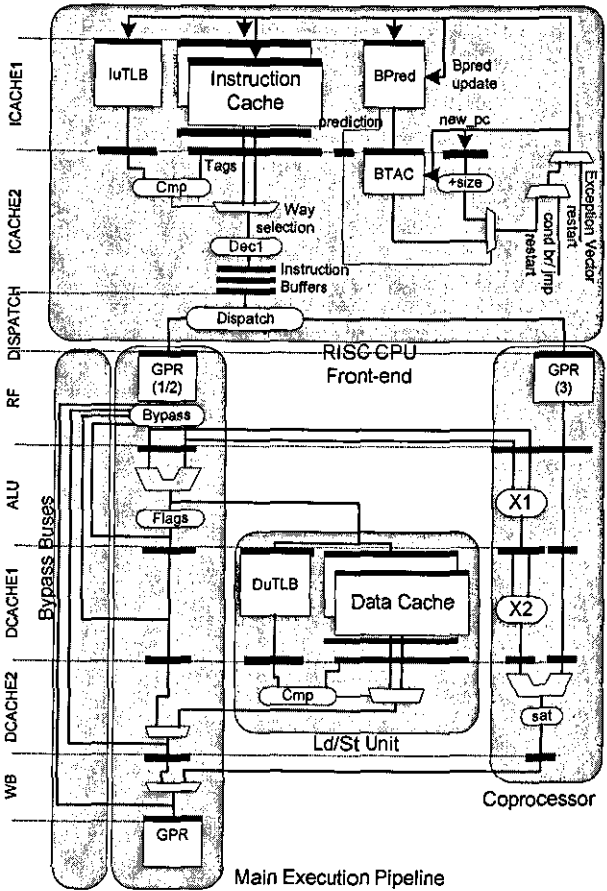


Figure 11: High-level schematic of limited dual-issue CPU

VIII. CONCLUSIONS AND FUTURE WORK

We utilized a combination of techniques to profile and optimize the ITU-T G729A and G723.1 speech coders. A further significant source of optimization lies with tapping the amount of data-level parallelism available in the workloads. Our group currently investigates vector architectures for the efficient execution of the speech coders.

Additional insight on the cycle effects will be provided through the cycle-accurate modeling of both coprocessors when attached to a more generic RISC CPU with limited dual-issue ability. This is portrayed in figure 11 where a high-performance scalar RISC processor with 8 pipeline stages and limited dual-issue capability (one scalar, one coprocessor) is described. This will allow for experimentation of the processor/co-processor design space and provide insight into the necessary microarchitecture requirements for the efficient execution of the workloads.

Finally, we are building the RTL model of the microarchitecture of figure 6 in the context of the system of figure 10.

REFERENCES

- [1] ITU-T Recommendation G.723.1, 'Dual Rate Speech coder for multimedia communications transmitting at 5.3 and 6.3 kbits/s', 3/96
- [2] ITU-T Recommendation G.729, 'Coding of speech at 8 kbits/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)', 3/96
- [3] M. Prasad, P. Arcy, M. Diamondstein, H. Srinivas, 'Half-Rate GSM Vocoder Implementation on a Dual-Mac Digital Signal Processor', Proceedings of the 1997 IEEE International Conference on Acoustics, Speech and Signal Processing, pp 619-622
- [4] Vinod Kathail, Shail Aditya, Robert Schreiber, B. Ramakrishna Rau, Darren C. Cronquist, Mukund Sivaraman, 'PICO: Automatically designing custom computers', IEEE Computer, 35(9), September 2002
- [5] D. Burger, T. Austin, 'Evaluating Future Microprocessors: The SimpleScalar Tool Set' <http://www.simplescalar.com>
- [6] V. A. Chouliaras, J. L. Nunez, "A scalar coprocessor for accelerating the G723.1 and G729A speech coders", accepted for publication in the IEEE International Conference on Consumer Electronics (ICCE03)
- [7] Y. Won, S. Sahni, Y. El-Ziq, 'A hardware accelerator for maze routing', IEEE Trans on Computers, vol. 39, no. 1, pp. 141-145, Jan. 1990
- [8] R. Cox, 'Three new speech coders from the ITU cover a range of applications', IEEE Communications magazine, pp. 40-47, Sept 1997
- [9] R. Cox, P. Kroon, 'Low bit-rate speech coders for multimedia communication', IEEE Communications magazine, pp.34-41, December 1996
- [10] 'The Leon-2 processor User's manual, XST edition, ver. 1.0.14', www.gaisler.com
- [11] 'AMBA Specification (Rev 2.0)', www.arm.com
- [12] A. Royo, J. Moran, C. Lopez, "Design and implementation of a coprocessor for cryptography applications", Proceedings of the 1997 IEEE European Design and Test Conference (ED&TC'97), pp 213-217
- [13] B. Costinescu, R. Ungureanu, M. Stoica, E. Medve, R. Pread, M. Alexiu, C. Ilas, 'ITU-T G729 Implementation on Starcore SC140', AN2094/D, Rev. 0,02/2001, www.motorola.com
- [14] S. Chang, J. Hu, 'Real-time implementation of G723.1 speech codec on a 16-bit DSP processor', Department of electronic and control engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C
- [15] M. Soler, A. Andre, E. Closse, J. Laval, F. Balestro, D. Morche, P. Senn, 'An embedded DSP platform for multi-standard ITU G728, G729 & G723.1 audio compression', France Telecom, CNET
- [16] M. Medina, G. Ezer, P. Konas, 'Verification of configurable processor cores', proceedings of the 2000 Design Automation Conference, Los Angeles, California
- [17] 'The Sparc Architecture Manual Version 8', www.sparc.com
- [18] A. Wang, E. Killian, D. Maydan, C. Rown, 'Hardware/software instruction set configurability for system-on-chip processors', proceedings of the 2001 Design Automation Conference, Las Vegas, Nevada
- [19] W. Raab, N. Bruels, U. Hachmann, J. Harnisch, U. Ramacher, C. Sauer, A. Techmer, 'A 100-GOPS programmable processor for vehicle vision systems', IEEE Design and Test of Computers, pp.8-16, Jan-Feb 2003
- [20] Arithmetic module generator, <http://www.fysel.ntnu.no/modgen/>
- [21] A. S. Spanias, 'Speech Coding: A tutorial review', Proceedings of the IEEE, vol. 82, no. 10, pp.1541-1581, October 1994
- [22] Y. Zhao, A. Wang, M. Moskewicz, C. Madigan, 'Matching architecture to application via configurable processors. A case study with the Boolean satisfiability problem', proceedings of the 2001 International Conference on Computer Design: VLSI in Computers and Processors

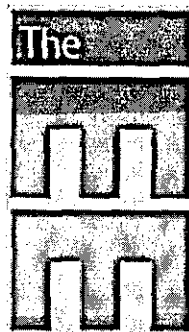


Vassilios A. Chouliaras was born in Athens, Greece in 1969. He received a B.Sc. in Physics and Laser Science from Heriot-Watt University, Edinburgh in 1993 and an M.Sc. in VLSI Systems Engineering from UMIST in 1995. He worked as an ASIC design engineer for Intracom SA and as a senior R&D Engineer/Microprocessor architect for ARC International. Currently, he is a lecturer in the Department of Electronic and Electrical Engineering at the University of Loughborough, UK. His research interests include superscalar and vector CPU microarchitecture, high-performance embedded CPU implementations, performance modeling, custom instruction set design and self-timed design.



José Luis Núñez is a research fellow in the department of Electronic Engineering at Loughborough University where he has worked since 1997. His current interests include the areas of lossless data compression, reconfigurable vector architectures, FPGA-based design and high-speed data networks. He received his BS and MS degree in Electronics Engineering from Universidad de La Coruña (La Coruña, Spain) and Universidad Politécnica de Cataluña (Barcelona, Spain) respectively in 1993 and 1997. He received his PhD degree at Loughborough University (Loughborough, England) in 2001 working in the area of hardware architectures for high-speed data compression.

Paper PJ2: V. A. Chouliaras, J. L. Nunez, K. Koutsomyti, S. R. Parr, D. J. Mulvaney, S. Datta, '*On the development of a custom vector accelerator for high-performance speech coding*', IEE Electronic Letters, Vol. 40, Issue 24, 25 Nov. 2004, pg 1559-1561



Manuscript for Review

On the development of a custom vector accelerator for high-performance speech coding

Journal:	<i>Electronics Letters</i>
Manuscript ID:	draft
Manuscript Type:	Letter
Date Submitted by the Author:	n/a
Complete List of Authors:	Chouliaras, Vassilios; University of Loughborough, Electronic and Electrical Engineering Nunez-Yanez, Jose; University of Loughborough, Electronic and Electrical Engineering Koutsomyti, Konstantia; University of Loughborough, Electronic and Electrical Engineering Parr, Simon; ; University of Loughborough, Electronic and Electrical Engineering Mulvaney, David; University of Loughborough, Electronic and Electrical Engineering Datta, S.; University of Loughborough, Electronic and Electrical Engineering
Keywords:	VECTOR PROCESSOR SYSTEMS, VERY HIGH SPEED INTEGRATED CIRCUITS, COMPUTER ARCHITECTURE

On the development of a custom vector accelerator for high-performance speech coding

V. A. Chouliaras, J. L. Nunez, K. Koutsomyti, S. R. Parr, D. J. Mulvaney, S.
Datta

The addition of custom vector instructions to the G.729A speech coding algorithm is shown to significantly reduce its computational complexity. The identified vector extensions are implemented in the form of a configurable vector accelerator, tightly coupled to a 32-bit Sparc V8-compliant reduced instruction set (RISC) processor. Architectural simulation demonstrates that a reduction in complexity of up to 60%, for a vector length of sixteen 16-bit elements, is achievable in current very large scale integration (VLSI) technology.

Introduction: The G.729A standard speech coding algorithm, as recommended by the International Telecommunications Union (ITU) [1], is a reduced complexity version of the Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP) coder of the G.729 recommendation. This coder belongs to the time domain Analysis-by-Synthesis (AbS) class of speech coders. Such coding schemes have been widely adopted as they produce speech that is subjectively of high quality while maintaining a low transmission rate. In the AbS approach, the encoder (analysis) incorporates the decoder (synthesis) to determine the initial excitation signal and uses

linear prediction techniques to determine the coefficients of the speech synthesis filter. In the CS-ACELP coder the initial excitation for the synthesis filter is obtained from two codebooks. An adaptive codebook is used to model an estimated pitch period that represent the voice sounds originating in the vibrating vocal chords and a fixed codebook is used to model unvoiced sounds such as nasal or plosive utterances. The excitation signal is then applied to a tenth-order synthesis filter whose transfer function models the human vocal tract. The coefficients of this synthesis filter are obtained by applying linear prediction analysis to the original speech input. The residual error between the reconstructed speech produced by the synthesis filter and the original input speech is processed by a perceptual weighting filter. The output of the filter is matched against the adaptive codebook elements to determine both the codebook index and gain that best approximate the residual signal. The codebook contribution is removed from the residual and a new match is made using the fixed codebook. The index and gains for both codebooks are assembled together with the synthesis filter coefficients to form the bitstream transmitted to the decoder. This entire processing is repeated for every 10ms frame of the voice signal. At the receiver, the bitstream is disassembled to obtain the filter coefficients and the codebook parameters. The excitation is constructed by adding the adaptive and fixed codebook vectors scaled by their gains and it is then filtered through the same synthesis filter used during encoding. Additional post-processing of the speech signal is performed to enhance its quality.

Methodology. The ITU G.729A reference code was profiled both in native mode (Intel x86) and on the SimpleScalar [2] tools to ensure consistency and general applicability of results. The SimpleScalar environment is a complete

computer architecture modelling tool based around a simulated 32-bit MIPS-II type processor with 64-bit opcodes. The compiler was GCC 2.7.3 with optimizations (-O3).

The respective complexity metrics (real time for native mode profiling, dynamic instruction count for the simulated processor) of the G.729A encoder were within 5% of one another despite the fundamentally different instruction-set architectures (ISAs). Our experiments therefore concentrated on the simulated infrastructure as this produces results that are both independent of the sampling issues of the native profiling tool and more close to real implementations of RISC/DSP (digital signal processor) processing kernels for telecommunication applications. In our previous work we quantified on the relative complexity of the DSP emulation instructions for the G723.1 and G729A ITU reference implementations and proposed two scalar accelerators [3 4] to reduce that complexity by up to 69% and 65% respectively. That complexity distribution is presented in Fig. 1. Subsequent code reviews revealed that significant data-level parallelism (DLP) exists in the workload resulting in the architectural definition of vector extension instructions based on the DSP emulation instructions and their associated implementation as a vector accelerator. The Data-parallel sections of the coder were re-written in vector assembly with the vector instructions used in place of the inefficient C implementation.

The processor state of the vector accelerator is depicted in Fig. 2. It consists of sixteen vector registers of statically-configurable length, two vector accumulators, two vector mask (predicate) registers and sixteen 32-bit scalar registers. The proposed vector ISA consists of fixed-point arithmetic, multiply-add, shift (with negative shift capability), mask processing, merge, vector load

and store instructions in 16-bit and 32-bit variants.

Microarchitecture: The vector extensions are implemented as a tightly-coupled coprocessor attached to a high-performance, 32-bit configurable processor [5]. The combined microarchitecture is shown in Fig. 3. Instructions are fetched from the multi-way set-associative instruction cache and clocked in the instruction register. When a vector opcode is identified, the source operand addresses are extracted and passed to the synchronous vector register file. Vector register access is followed by operand bypassing in both the scalar and vector pipelines. The EXEC stage is the first phase of execution of the vector ISA and the only execution stage in the main scalar pipeline. In addition, vector load operations return their data to the vector pipeline at the end of the datapath. Intermediate results are pipelined to the next stage (DMEM/EXEC2) for the final phase of vector execution. During this stage, scalar operands return to the main RISC pipeline via the data cache load path. Finally, results commit to the vector register file after being stored in a staging register. The staging register is necessary for performance reasons, relating to the set-up time of the register file SRAMs, and pipeline symmetry, for the precise processor state recovery following an exception.

Results: The vectorized workload was executed with vector extensions enabled and the dynamic instruction count (complexity) was measured for all the ITU test vectors and for vector lengths ranging from 2 (32 bits) to 128 (1024 bits). The normalized complexity of the vectorized workload for all input vectors and vector lengths is shown in Fig. 3. It is clear that significant reductions in complexity are achieved at vector lengths in the range 2 to 16,

corresponding to a range of vector datapath widths from 32 bits to 512 bits. Such widths are realizable in current VLSI technologies. Little further improvement in complexity occurs for vector lengths greater than 16, but local minima do occur at vector lengths of 32, 64 and 128 (not shown). Configurations with vector lengths greater than 16 are unrealistic in practice due to the significant silicon overhead incurred by such wide datapaths and the need for very long cache fill bursts. Such configurations were investigated in this study only for completeness. The results of Fig. 3 reflect a shared multiplier resource per two 16-bit elements. Our preliminary investigation shows that a dedicated multiplier per 16-bit element provides an additional benefit of the order of 1% only.

Conclusion. A custom vector ISA was developed that offers significant reduction in the complexity of the G.729A speech coder. Initial architectural results are very promising, demonstrating a reduction in algorithmic complexity of up to 60% that can be realized in current VLSI implementations. Further work will focus on the combining the scalar extensions reported earlier and the vector extensions reported here, for both G.729A and alternative speech coding standards.

References

1. ITU-T Recommendation G.729, '*Coding of speech at 8 kbits/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)*', 3/96, Place des Nations, CH-1211, Geneva, Switzerland.
2. D. Burger, T. Austin, '*Evaluating Future Microprocessors: The SimpleScalar Tool Set*' <http://www.simplescalar.com>
3. V. A. Chouliaras, J. L. Nunez, '*A scalar coprocessor for accelerating the G723.1 and G729A speech coders*', Proceedings of the IEEE International Conference on Consumer Electronics (ICCE03), Los Angeles, California, USA
4. V. A. Chouliaras, J. L. Nunez, 'Scalar Coprocessors for accelerating the

G723.1 and G729A Speech Coders', IEEE Transactions on Consumer Electronics, Vol. 49, Issue 3, Aug. 2003, pg. 703-710.

5. 'The Leon-2 processor User's manual, XST edition, ver. 1.0.14', www.gaisler.com

Acknowledgments

This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under contact GR/S44976/01

Author's affiliations:

V. A. Chouliaras, J. L. Nunez, K. Koutsomyti, S. R. Parr, D. J. Mulvaney, and S. Datta are with the Department of Electronic and Electrical Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK.
Email: v.a.chouliaras@lboro.ac.uk

Figure Captions

Fig. 1: ITU G.729A reference implementation profiling of the DSP emulation instructions

Fig. 2: Vector accelerator state

Fig. 3: Relative complexity reduction of the vectorized G.729A encoder

Fig. 4: Microarchitecture of the tightly-coupled vector accelerator and the main scalar processor

Figure 1

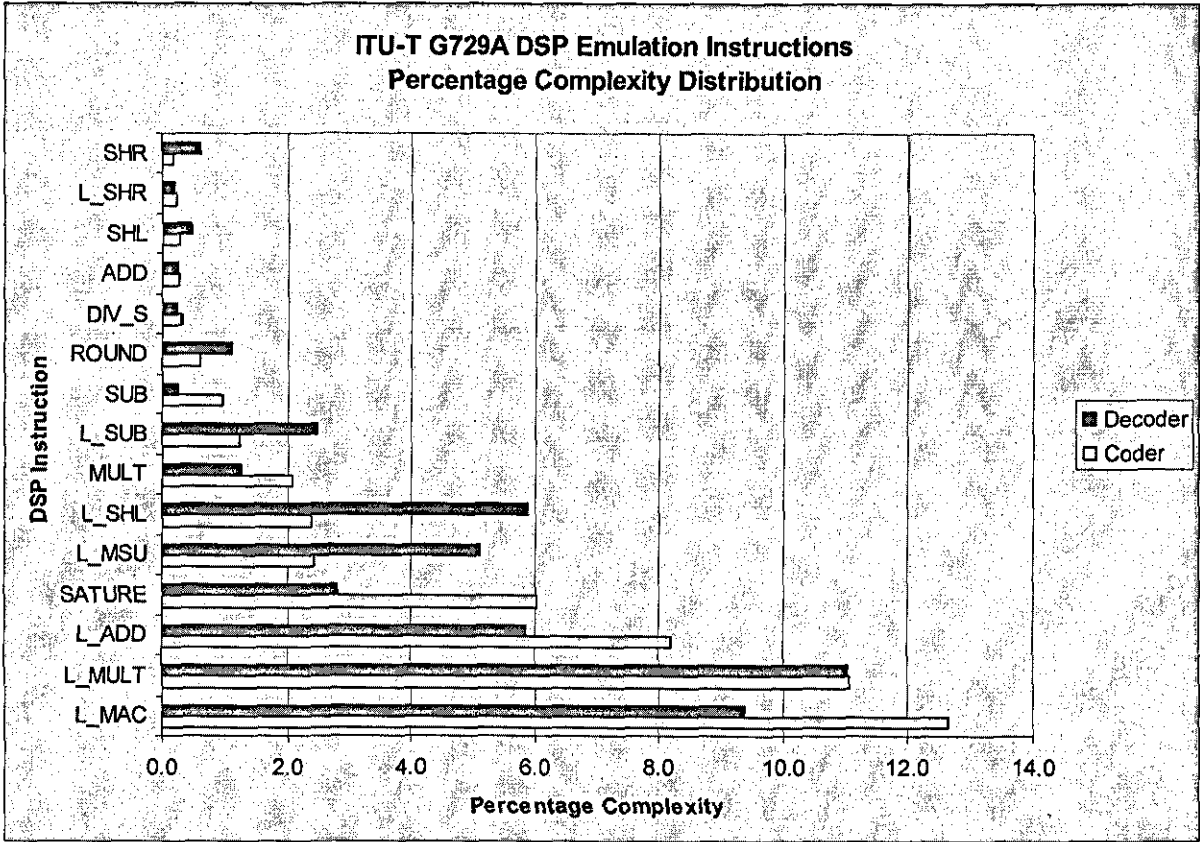


Figure 2

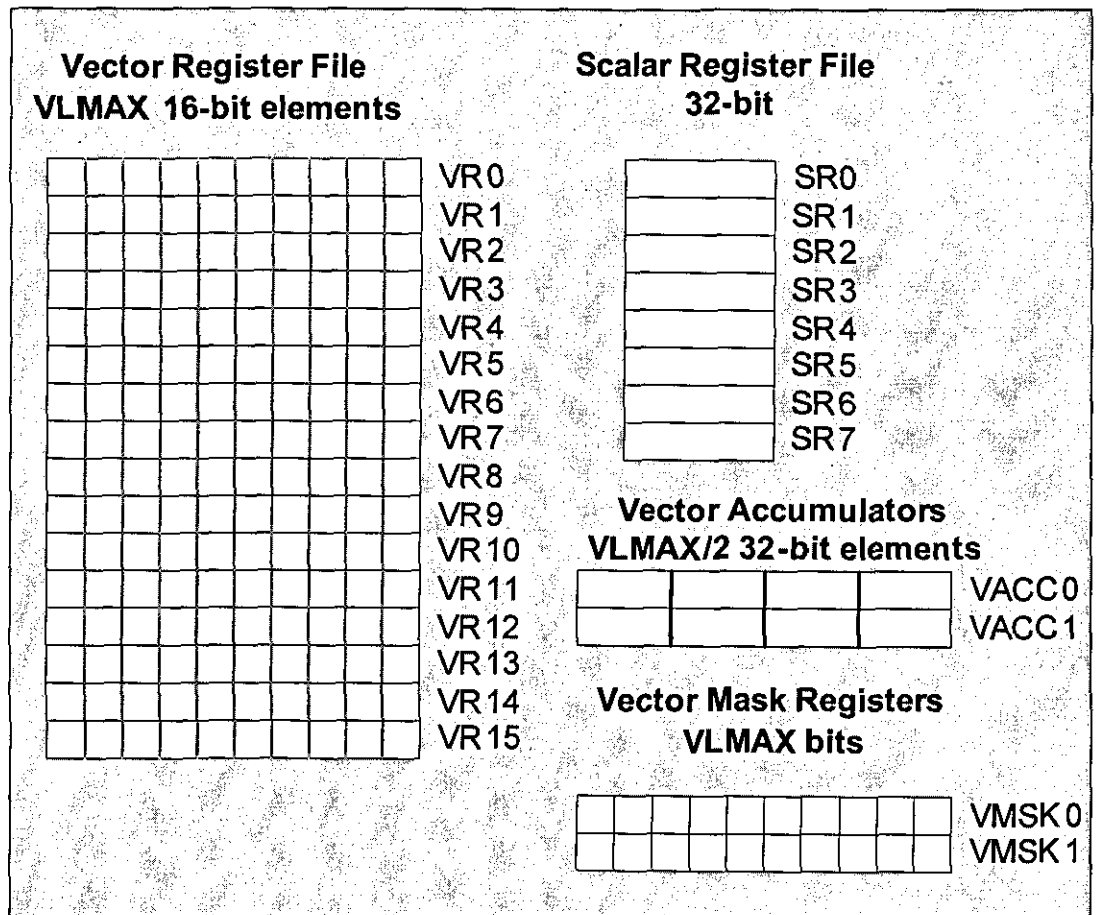


Figure 3

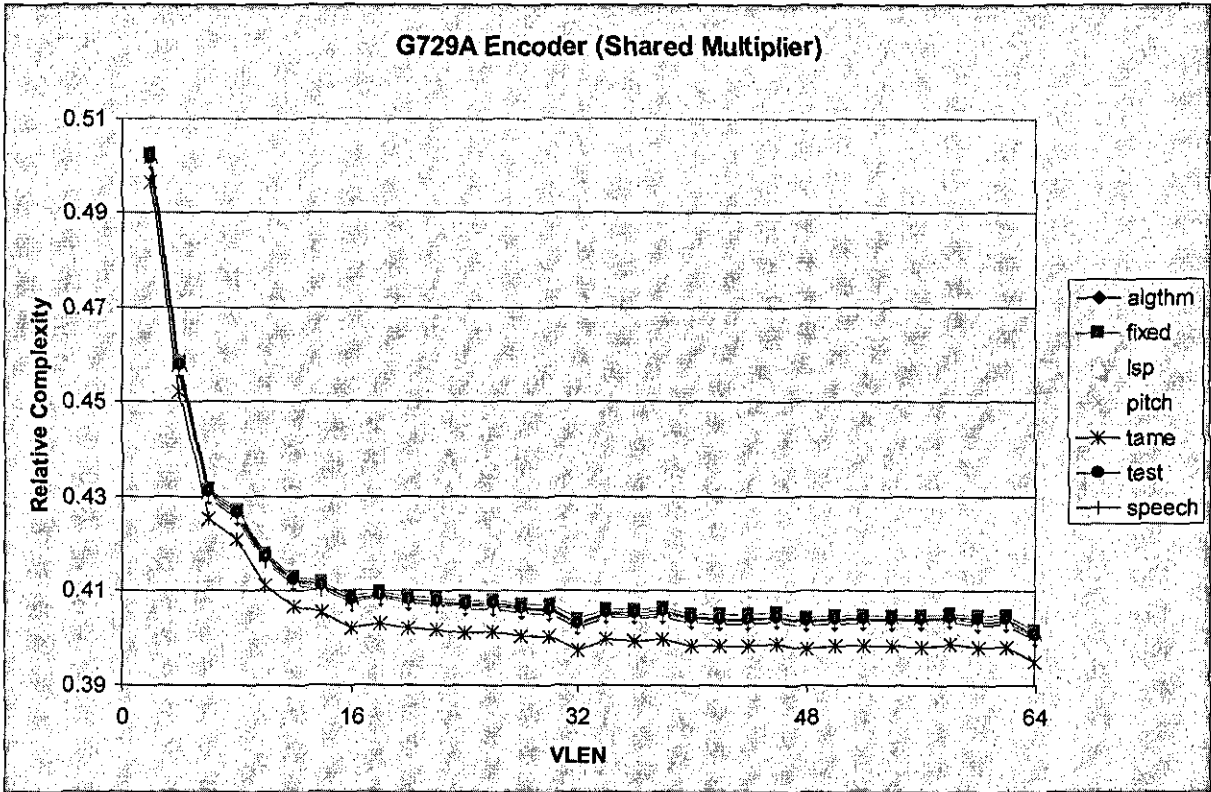
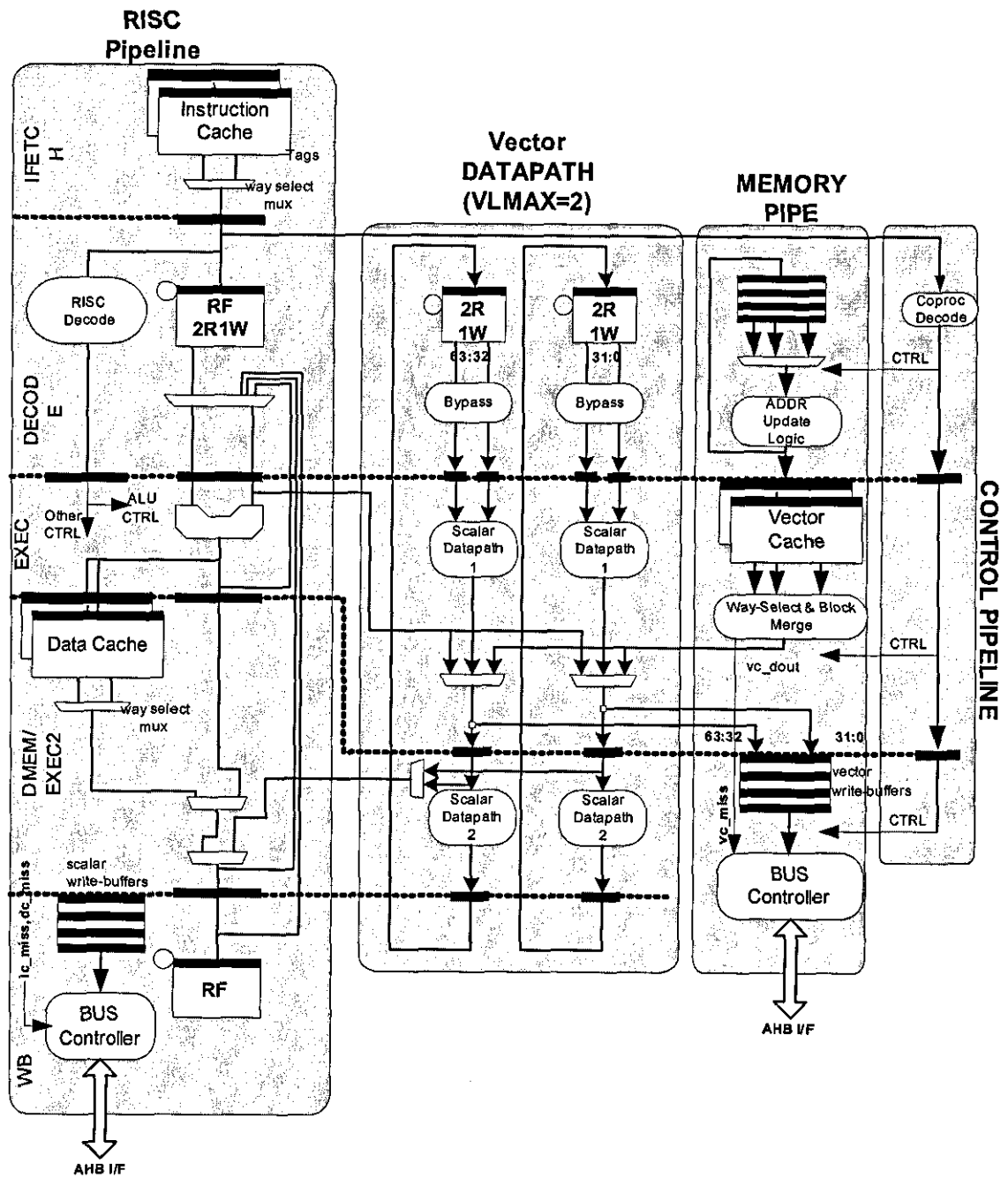


Figure 4



Paper PJ3: **V. A. Chouliaras** J. L. Nunez, D. J. Mulvaney, F. Rovati, D. Alfonso, '*A Multi-standard Video coding accelerator based on a vector architecture*', IEEE Transactions on Consumer Electronics, Vol. 51, Issue 1, Feb 2005, pg 160-167

A Multi-Standard Video Accelerator based on a Vector Architecture

V. A. Chouliaras, J. L. Nunez, D. J. Mulvaney, F. S. Rovati and D. Alfonso

Abstract — *A multi-standard video encoding coprocessor is presented that efficiently accelerates MPEG-2, MPEG-4 (XViD) and a proprietary H.264 encoder. The proposed architecture attaches to a configurable, extensible RISC CPU to form a highly efficient solution to the computational complexity of current and emerging video coding standards. A subset of the ISA has been implemented as a VLSI macrocell for a high performance 0.13 μm silicon process¹.*

Index Terms — MPEG2, MPEG4, H264, vector/SIMD accelerator, embedded RISC

I. INTRODUCTION

The past 10 years have witnessed an explosion in the quantity of visual information that must be transmitted and stored efficiently using limited and expensive resources. Advanced video coding allows orders of magnitude reduction in the required bit-rates and is regarded as an enabling technology in moving personal communications to a higher level of interactivity. This technology is being deployed particularly successfully in embedded systems for personal use such as Personal Digital Assistants (PDAs), digital cameras, palmtop computers and cellular phones as well as portable video game consoles and DVD players [1]. The major requirement of such systems is high-quality, low bit-rate video coding implemented in high-performance and low-power VLSI systems. Applications such as DVD decoding have so far been implemented in hardware platforms consisting of one or more embedded CPUs with associated accelerators for targeting the computationally complex functions. However, forthcoming applications such as personal wireless video involve compute-intensive real-time coding as well as decoding and this increase in the required computational capability can only be met by specialized video architectures.

Standard mobile wireless networks based on 2G GSM technology operate at 9.6 Kbit/s bandwidth and are thus unsuitable to support quality, real-time video. 2.5G GPRS provides an interim solution supporting 115 Kbit/s whereas upcoming 3G technology is expected to achieve at least 144

Kbit/s on fast moving stations, rising to 384 Kbit/s for pedestrian or slow moving stations. These figures indicate that video transmission must be achieved at less than 100 Kbit/s in emerging wireless networks to become as successful as current voice telephony

II. BACKGROUND

Substantial research has been conducted at the algorithmic level with the aim of developing improved video codecs capable of achieving high quality video encoding at lower bit rates than existing standards. Typically, lower bit-rates and higher PSNR values are achieved by sophisticated techniques that exploit the spatial and temporal redundancy present within the picture frame (intra-frame) and across frames (inter-frame) in a video sequence. Intra-frame coding removes the spatial redundancy within a single frame and the techniques utilized for this task are derivatives of those used for still image coding while inter-frame coding removes the temporal redundancy by coding the difference between the frames of a video sequence. Three fundamental video coding methods can be identified: those based on the discrete cosine transform (DCT) [2], [3], those employing the wavelet transform [4] and those adopting fractal-based coding algorithms [5]. The DCT-based methods are currently much more popular than the other two methods and form the basis of all the current international standards in digital video coding. The very high computing requirements of these multimedia workloads have created a demand for high-performance execution engines which can be categorized into four major microarchitectural approaches namely very-large-instruction-word (VLIW), superscalar, vector-based and application-specific.

Traditionally, Digital Signal Processors have been used for high-performance signal processing tasks such as audio, speech and video coding. A state-of-the-art DSP device [6] uses a wide VLIW architecture to execute up to 8 instructions per cycle at a maximum frequency of 1 GHz. Taking the VLIW paradigm to the extreme, the licensable Silicon IP core in [7] is a highly-parallel (up to 60 instructions) solution targeted towards streaming embedded applications. Both architectures represent modern approaches to long-instruction-word (LIW) signal processing with the first device being a capable execution engine for mains-powered media and telecommunication applications and the later approach offering excellent acceleration capability to small-size applications (kernels). Neither architecture is optimized for mobile video however the ULIW core presents a very potent customization target.

¹ V. A. Chouliaras and D. J. Mulvaney are with the Electronics Systems Design Group, Department of electronic and Electrical Engineering, University of Loughborough, UK (V.A.Chouliaras@lboro.ac.uk, D.J.Mulvaney@lboro.ac.uk).

J. L. Nunez is with the Department of Electronic Engineering, University of Bristol, UK (J.L.Nunez-Yanez@bristol.ac.uk)

F. Rovati and D. Alfonso are with the Advanced System Technology Labs, STMicroelectronics, Agrate, Italy (fabrizio.rovati@st.com, danielle.alfonso@st.com)

In the category of high-performance desktop superscalar processors, there has been a universal adoption of multimedia-enhanced instruction set architectures (ISAs), based on short vector/SIMD support [8], [9]. Such extensions proved decisive in achieving real-time video encoding/decoding on a desktop workstation however, their power consumption and size renders such devices inappropriate for embedded applications.

A number of specialized DSP vendors have realized the abundance of data level parallelism in multimedia workloads and introduced architectures with higher levels of specialization for video coding. The video engine in [10] uses a VLIW core combined with SIMD support (up to 128 bit wide). It can issue up to 4 instructions per second while clocking at 300 MHz and taps parallelism both at the instruction and the data level. Another approach combines a 64-bit MIPS RISC processor with two hardwired units for motion estimation and run-length coding plus a vector-based 64-bit wide macro-block engine to extract data level parallelism from the rest of the functions [11]. The parallel video DSP chip in [12] uses a 32-bit RISC processor, a 512-bit vector architecture and dedicated motion estimation coprocessors to achieve 3 billion multiply-accumulate operations per second while clocking at a modest 100 MHz. This is a highly parallel, complex device capable of extracting significant data level parallelism in both standard and proprietary video codecs. The chip uses very wide vectors and achieves leading-edge performance, clearly demonstrating the full potential of a dedicated vector architecture.

In the application-specific category, a number of commercial ASIC and FPGA-based hardware solutions targeting video coding are currently available based on the paradigm of combining a general purpose embedded microprocessor with custom hardware for the compute-intensive tasks. The Video Encoder in [13] targets MPEG-4 simple profile encoding. It includes a number of hardwired units to accelerate DCT coding, motion estimation, quantization and bit stream packing. The multimedia coprocessor in [14] combines a standard 32-bit ARM9 RISC processor with custom blocks to accelerate the most compute-intensive tasks such as motion estimation. The audio/video encoder chip in [15] is based on a MIPS-like RISC processor with DSP extensions for audio/video encoding. Finally, the MPEG-2 encoder in [16] is available as a silicon IP core targeting high-performance field-programmable gate arrays (FPGA). Its architecture incorporates macro-block processing engines for motion estimation, forward DCT and quantization. These solutions offer competitive performance at the expense of flexibility, scalability and their ability to handle emerging standards.

This work identifies vector architectures as the most potent engines for tapping the abundant Data Level Parallelism in three major video coding standards namely, MPEG-2, MPEG-4 and H264 and proposes a *highly targeted, configurable, vector coprocessor* that can be attached to an embedded RISC CPU. This coprocessor is capable of delivering better area,

power and performance metrics compared to other programmable solutions, in order to meet the requirements of wireless embedded applications. This paper deals primarily with the performance aspect of this programmable vector accelerator.

III. VIDEO STANDARD REVIEW

A. MPEG-2

MPEG-2 is a very popular, lossy video compression standard currently employed in many consumer products such as DVD players, DVD recorders and digital set top boxes. This standard was introduced in 1994 by the ISO/ITU-T [17] to support high quality video at transmission rates ranging from 4 to 80 Mbit/s. The MPEG-2 codec is based on the DCT either of the residual data, obtained after performing motion estimation (ME) and compensation (MC) to remove redundancy between frames (inter-frame coding), or of the original luminance and chrominance data when removing redundancy within the same frame (intra-frame coding). These transformations are followed by quantization which removes the high spatial frequency components in order to significantly reduce the required channel rate while maintaining good visual quality. MPEG-2 has achieved a universal acceptance status and is the baseline video coding standard in this work.

B. MPEG-4

The MPEG-4 multimedia standard covers a broad spectrum of audio and video coding schemes and representations allowing for the efficient transmission and storage of audiovisual information. This work focuses only on the visual coding aspect of that standard. The main processing functions involved in producing MPEG-4 video content are: ME, MC, forward discrete cosine transform (FDCT), inverse discrete cosine transform (IDCT), quantization (Q) and variable length coding (VLC). These functions are applied at block (8x8 pels) or macroblock (16x16 pels) levels. The MPEG-4 implementation selected for our work is the open-source XViD [18] version which in its most recent stable version implements a simple profile (SP).

MC is typically the most compute-intensive part of block-based video coding algorithms. It is used in predictive coded frames (P frames) to estimate the amount of movement experienced by the blocks or macroblocks across consecutive frames. The XViD algorithm uses one or four motion vectors (MV) per macroblock and half-pel motion compensation precision depending on quality settings. There is one MV for each of the luminance (luma) components in the macroblock, while the two chrominance (chroma) component MVs are calculated as a mean of the luma MVs. To determine these MVs, XViD uses a Predictive Motion Vector (PMV) algorithm that selects a MV for a block using the MVs generated for the neighboring blocks in the same frame and one MV from the previous frame. The selected MV is then further refined using a size-adaptive square area to search for a better MV. The search process is based on the compute-

intensive sum-of-absolute-differences (SAD) method over a search range. The MV that produces the smallest SAD value is selected as the best. The search is not performed exhaustively and early termination is used to speed-up the algorithm when a suitable SAD threshold value has been reached. The size of the square area is selected adaptively by the PMV algorithm. The motion compensation process refines the MV using half-pel interpolation so the value of a pel is calculated as the mean of adjacent pel values and a rounding factor.

DCT is used in the coding of both Intra (I) and Predictive (P) frames. The FDCT is applied to I frames to transform the image data blocks from the spatial domain to the frequency domain. The spatial frequency coefficients are then quantized and high-frequency information is removed resulting in a reduction in data volume with minimal loss of quality. The IDCT recovers an approximation of the original temporal image data using the quantized coefficients. These functions operate in the same way with P frames, except the input to the function is the residual remaining following the subtraction of the motion estimated block from the actual block of image data. Both FDCT and IDCT in XViD are 32-bit integer precision functions that transform first the columns and then the rows of each of the image blocks. Quantization in the frequency domain is the lossy step in the algorithm since the high-frequency (high-detail) components in the frame tend to generate low-value coefficients which are rounded down to zero after this step. The XViD algorithm selects the same quantization strategy as that recommended by ITU-T for the H.263 standard [19]. The quantization matrix is optimized for low-bit rate settings and has the same effect on high and low frequency coefficients.

VLC is the last stage of the video coding process and reduces the number of bits used for the frequency coefficients. XViD uses a (*last, run, level*) triplet format where the first bit indicates if the last non-zero coefficient is being coded, the run indicates how many zero coefficients precede the current non-zero coefficient and the level indicates the value of the current non-zero coefficient. To further speed-up this process the codes are stored in look up tables (LUTs) and the (*last, run, level*) triplets are used as addresses to these LUTs. This implementation is geared towards speed since it is clear that more sophisticated coding strategies are possible. VLC is in general a sequential process and is the only functional block where vector extensions do not offer any advantage.

C. H.264

H.264 is a hybrid video coding standard, developed by the Joint Video Team (JVT) of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Pictures Experts Group (MPEG) [20], [21]. With respect to its ancestors, it offers about 50% better compression while not compromising quality, due to a range of improvements that impact all aspects of the digital video encoding process [22].

Principal innovations of H.264 are *Multiframe Prediction*, which allows the use of more than one previous-frame as

reference for ME, sub-sample interpolation at $1/4^{\text{th}}$ of a pixel, and Macroblock partitioning which admits up to 16 independent MVs associated with each 16x16 pixels Macroblock. The Hadamard Transform is used to compute the SAD in the frequency domain and the classical 8x8 pixels DCT is replaced by a simpler integer version operating on 16x16 and 4x4 pixel blocks. Two different entropy coding schemes may be used: CAVLC (Context-Adaptive Variable Length Coding), a Huffman-like method, and CABAC (Context-Adaptive Binary Arithmetic Coding), a complex but high-performance variation of arithmetic coding.

The proprietary software implementation of H.264 used in this work includes a fast ME method, called 'Openslim', which is based on the correlation existing among spatially adjacent and temporally consecutive motion vectors. It operates in two steps, first selecting the best MV from a set of candidates chosen from vectors already computed for Macroblocks in the current and previous frame; second, testing a fixed number of displacement vectors around the best position found in the previous step. This algorithm executes two interleaved motion searches: a coarse search following the picture display order, and a fine search in the picture coding order; the latter exploiting the results of the former to achieve higher precision. Openslim provides performance almost identical to that of the common Full-Search Block-Matching, at only a very small fraction of the computation.

The bit-rate of the encoded sequences is controlled by an algorithm proposed by the JVT committee [23], which is a Constant Bit-rate Controller (CBR) derived by the classical TM5 method developed for MPEG-2 by the MPEG Software Simulation Group [24].

IV. PROBLEM FORMULATION

The aim of this research is to enable real-time video encoding in portable, wireless products through a combination of advanced hardware platforms executing explicitly parallelized (vectorized) versions of the three video coding standards. The proposed platform is based on an open source configurable, extensible, 32-bit Sparc V8-compliant [25] RISC CPU [26] augmented by a configurable vector accelerator.

All three standards were initially profiled unmodified on our default Instruction Set Simulator (ISS) which is based on the SimpleScalar Toolset [27]. The major complexity contributors identified at the function level are depicted in Fig. 1 (MPEG-2), Fig. 2 (MPEG-4) and Fig. 3 (H.264). In the case of MPEG-2, the major dynamic instruction count contributors were the inner loop of the ME function (DIST1), which computes the error of the current macroblock over an arbitrary reference macroblock. This function is called for all macroblocks in the search window of the reference frame and is independent of the search algorithm utilized.

For the case of MPEG-4, functions of major complexity are ME/MC, FDCT and te IDCT and finally Q. The complexity of these functions accounts for more than 80% of total complexity and the existence of significant amounts of data level parallelism

makes them very good candidates for vectorization.

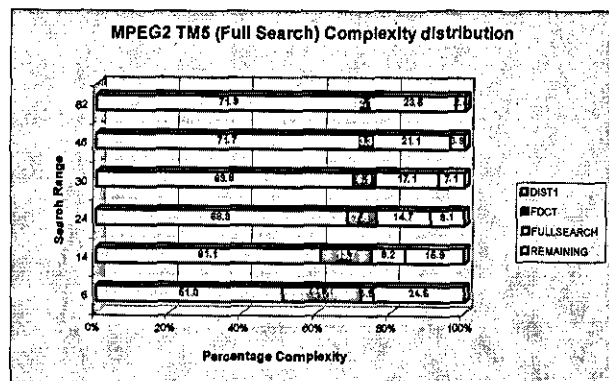


Figure 1: MPEG-2 TM5 complexity distribution

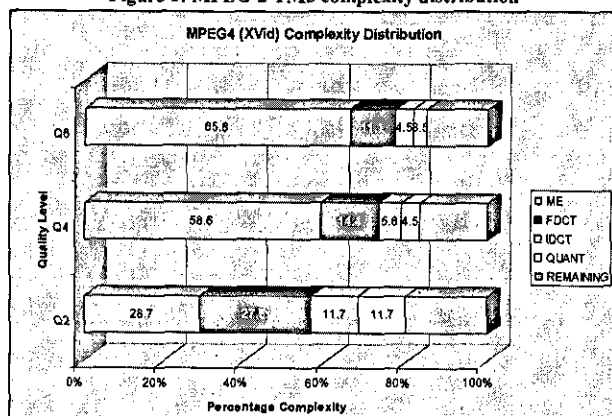


Figure 2: MPEG-4 (XViD) Complexity

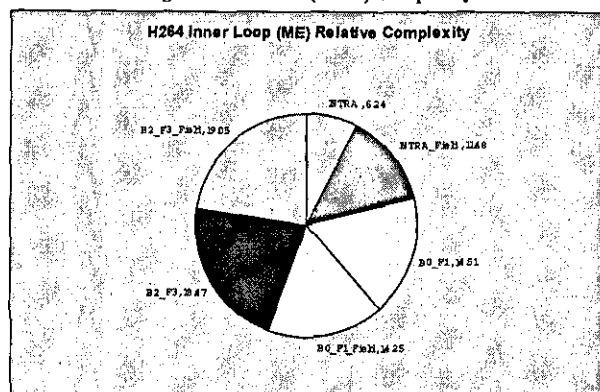


Figure 3: H.264 inner ME loop relative complexity (Non-Hadamard, Openslim ME)

Fig 3. depicts the relative complexity of the *motion estimation* function in the proprietary H.264 video coder, for different configurations: Bx is the number of B-frames between anchor frames, Fy indicates the number of previous reference frames used by Multiframe Prediction, Field refers to interlaced coding, while INTRA means that all the pictures were encoded as I-type, without ME. In INTRA_field coding, the top field of each picture is Intra Coded, while the bottom field is predicted from the top.

ARCHITECTURAL RESULTS

Following profiling all three video coding standards were vectorized and, in the process, a common vector Instruction Set Architecture (ISA) was developed. The vector extension instructions were added to our default ISS and the performance (dynamic instruction count) evaluated over vector length, search range, and ME algorithm. Results were obtained for multiple video sequences in the case of MPEG-2 and MPEG-4.

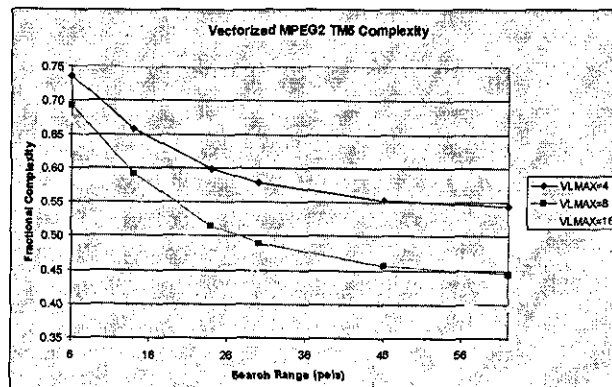


Figure 4: MPEG-2 TM5 fractional complexity

Fig. 4 shows the final complexity of the MPEG2 encoder as a function of vector length (4, 8 and 16 bytes) and for a search range of up to 63 pels. A more detailed account of the MPEG-2 aspect of this work can be found in [28].

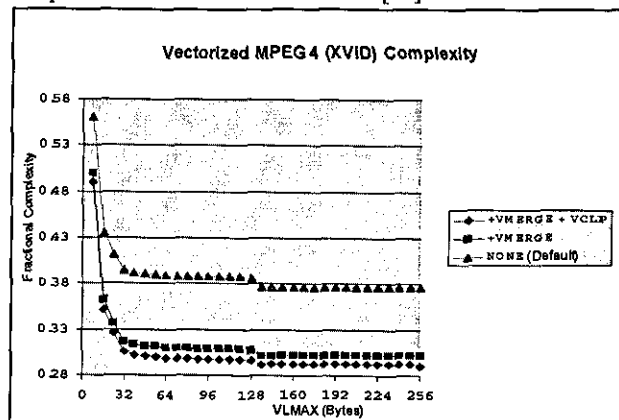


Figure 5: MPEG-4 (XViD) fractional complexity

Fig. 5 shows the complexity of the vectorized MPEG4 encoder for three configurations: Default incorporates a basecase video ISA with the VMERGE and VMERGE+VCLIP curves showing the benefit of these two additional instructions. The maximum vector length ranges up to 256 bytes however, no additional benefit is achieved beyond 128 bytes with realistic VLSI implementations ranging between 16 to 64 bytes. Though not shown explicitly, the complexity of the XViD ME was reduced by 85% at a vector length of 24 bytes and the complexity of MC was reduced by 70% at the same vector length. DCT vectorization involved two functions for forward and inverse DCT. The 2-D 8x8 DCT is performed using 2 1-D DCTs that are applied first to the columns and then to the rows of the 8x8 pixel array. In order efficiently vectorize the DCT, extra functionality was needed to transpose the arrays so that vector load and vector store

instructions would perform unit-stride accesses. The complexity of the DCT functions after vectorization was reduced by 82% at 32-byte vector length. Finally, vectorization of the quantization functions yielded a reduction in complexity of the order of 93% at a vector length of 64 bytes and a 90% reduction at a vector length of 32 bytes.

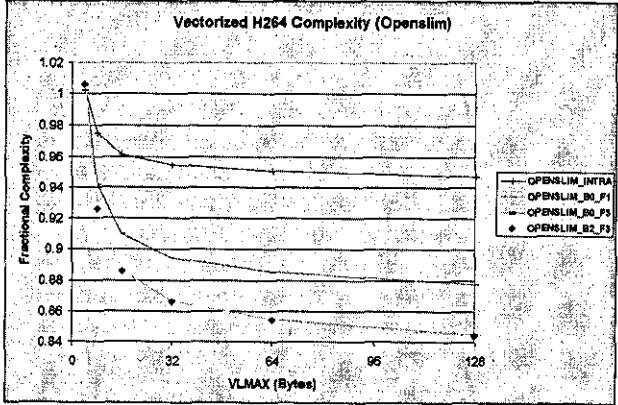


Figure 6: Vectorized H.264 Encoder Complexity (16 Search Range)
Fig. 6. depicts the complexity of the proprietary H264 implementation. We vectorized only the SAD computation for the Non-Hadamard transform case. The H.264 results are thus preliminary, with vectorization applied to the ENC_SATD function only (up to 72% complexity reduction within that function). The overall complexity of the vectorized encoder is approximately 86% for a vector length of 32 bytes and this is the subject of ongoing investigation.

VI. PROGRAMMERS MODEL AND INSTRUCTION

Fig. 3 shows the programmer-visible state of the parametric vector accelerator.

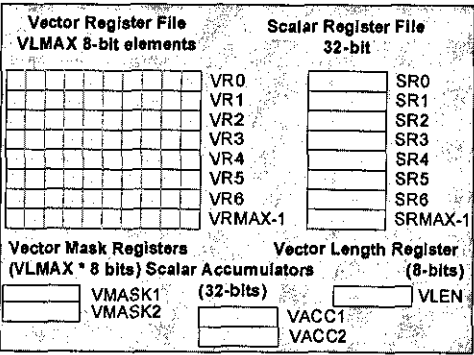


Figure 7: Accelerator Programmer's Model

The programmer-visible state is itself parameterized and split into three main register categories namely vector (up to VRMAX registers), scalar (up to SRMAX registers) and miscellaneous (mask, scalar accumulators and vector length register). The ISA consists of 45 custom instructions in five categories: Miscellaneous, load/store, cross-lane, video, and reduction operations. Miscellaneous operations affect the VLEN register and move scalar data between the main RISC processor and the vector coprocessor. The vector load/store instructions transfer scalar and vector operands from the vector load/store unit (VLSU) to the accelerator scalar and vector

register files respectively. All such operations support register-indirect and register-indirect with post-increment/pre-decrement addressing modes. The category of cross-lane operations includes two and three-operand byte-granularity permute, pack and unpack operations and inter-element shifts. Their major characteristic is that a particular scalar pipeline is able to access operands from all other scalar pipelines. The common datapath component is the triple-operand permute unit which allows for arbitrary, byte-wise permute operations of two source vectors under the control of a third source vector. All pack/unpack operations are special (hardwired) variations of the two or three operand permute which drive pre-computed patterns to the permute unit decode logic instead of a third vector source. Video operations cover the bulk of the vector ISA for video acceleration. They include 8, 16 and 32-bit intra-element shifts, vector compare-merge, vector multiply (8 and 16 bits), SAD and arithmetic operations. Finally, reduction operations compute a scalar value out of one or more vector operands and write it back to one of the accumulators or scalar registers.

VII. MICROARCHITECTURE

The accelerator attaches to the configurable, extensible Sparc-V8 compliant Leon2 CPU. Fig. 8 depicts a high-level schematic of the proposed AHB-based [29] SoC sub-system in which the major blocks are the Scalar CPU, the Coprocessor I/F, the video coprocessor and the high-speed external memory. A more detailed schematic of the core processor/coprocessor microarchitecture is depicted in Fig. 9.

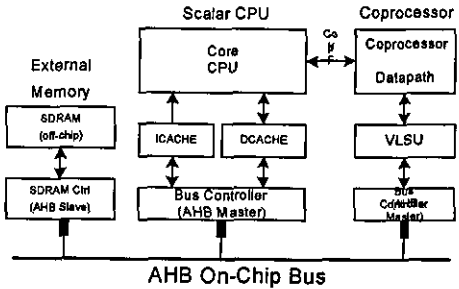


Figure 8: SoC Kernel

The Leon 2 CPU is a standard 5-stage RISC pipeline. Instructions are fetched from the multi-way, set-associative instruction cache and clocked into the instruction register. Decoding takes place in the DECODE stage with the RISC register file accessed at the falling edge of the clock. The bypassing logic in DECODE determines whether register file data or internally pipelined results are clocked in the ALU input registers. During EXEC, the ALU operation is performed and a virtual address is computed. Scalar data cache access takes place during DMEM/EXEC2 and scalar results return to the RISC pipeline during this cycle. Finally, results are clocked into an intermediate register prior to committing to the processor register file. The processor incorporates a configurable data cache in a write-through configuration with no-write-allocate policy. The scalar data cache forms part of

the data pipeline which includes a 3-word, non-collapsing write buffer serving to decouple the high-speed execution pipeline from the slower SoC memory subsystem. The core CPU finally includes a parametric instruction cache. It is a standard design and supports instruction streaming (processor operating in parallel to the refill sequence while the missed instruction/data word is fetched). Both caches are refilled over the AHB via the processor bus controller, which includes the ICache/DCache arbitration mechanism.

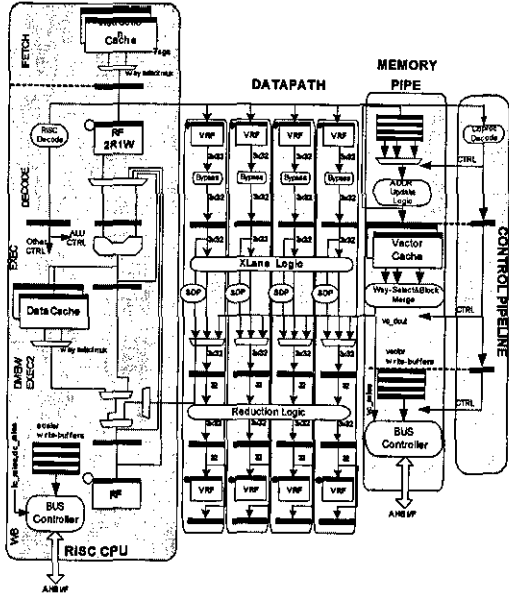


Figure 9: Processor-Coprocessor Microarchitecture

The vector coprocessor is a configurable, 4-stage pipelined microarchitecture and consists of the vector datapath, the vector memory pipeline and the control pipeline. The vector datapath consists of the vector register file, and bypass logic, the cross-lane logic, the SIMD datapaths and the reduction logic.

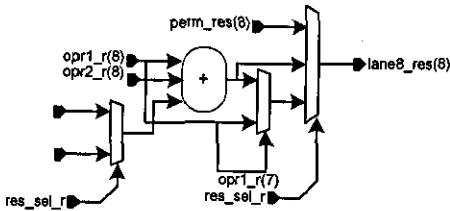


Figure 10: LANE8 microarchitecture

A major configuration parameter is the *maximum* vector register length in bytes, VLMAX, which dictates the geometry of the vector register file SRAMs and the number of scalar datapaths instantiated. The vector pipeline in particular is segmented into VLMAX/4 32-bit scalar lanes, each capable of operating on four bytes, two half-word or one word element per cycle (4-way SIMD). The logic of a single 8-bit datapath slice (LANE8) is depicted in Fig. 10.

Four LANE8 entities are composed as shown in Fig. 11 to form a 32-bit scalar lane. The logical hierarchy of the EXEC stage does not include the cross-lane logic since the later is shared across all datapaths in the EXEC stage.

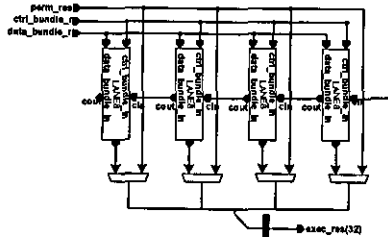


Figure 11: LANE32 microarchitecture

The vector register file supplies operands to the vector pipeline and is segmented into VLMAX/4 elements, each consisting of 3R1W, VRMAX-by-32-bit embedded RAMs with byte-write capability. Due to the unique requirements of the MPEG-2 algorithm, a special vector register file in a 5R1W configuration can be instantiated however, at the expense of limiting VRMAX to 8. A further parameter allows the use of latch, flop or dual-port compiled SRAM configurations. In the later case, there are two more possible configurations: the first makes use of three dual-port memories with common write port and individual read ports and the second instantiates two dual-port blocks with the read port of the first clocked at double the processor clock allowing for area/frequency tradeoffs.

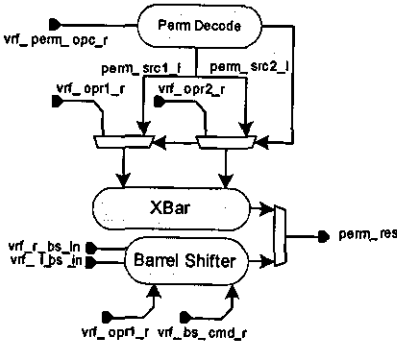


Figure 12: Cross-lane Logic

The cross-lane logic of Fig. 8 executes all permute, pack/unpack instructions and inter-element shifts². To eliminate unnecessary toggling in the crossbar when executing a non-permute operation, datapath-gating logic is instantiated which hardwires the crossbar vector operands to '0'.

A number of opcodes specify the add-reduction of a vector register to a single 32-bit value, writing that scalar result to a scalar address/data register. The reduction logic is situated in the DMEM/EXEC2 stage and Fig. 13 details a possible implementation of the add-reduction logic, for VLMAX=16 bytes (128 bits). The implementation depicted consists of log₂(16)+1=5 full-adder stages, each of increasing bit width. Other implementations are possible, depending on target technology and synthesis tool ability to perform advanced transformations of adder trees.

² The crossbar is potentially a source of concern in timing and routing closure for VLMAX>32 configurations due to accessing all vector operands from each scalar lane and producing a vector result fanning-out to all scalar lanes.

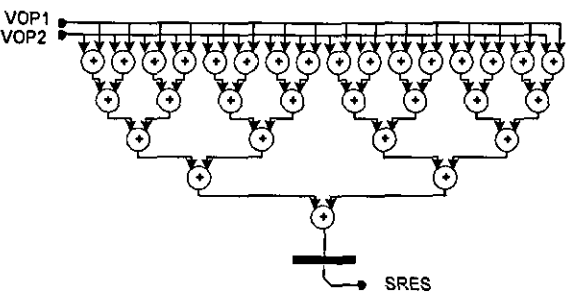


Figure 13: Parametric reduction logic (VLMAX=16)

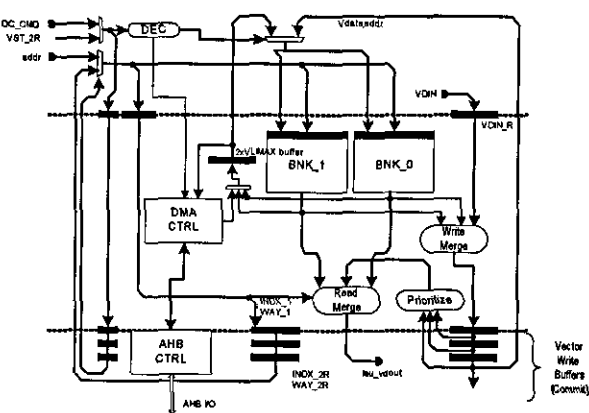


Figure 14: Local-RAM VLSU Microarchitecture

on a high-performance, 0.13 μm , 8-copper layer CMOS technology. The chosen configuration includes a 4-way, 8 KB instruction cache with a 16-byte block length, a 4-way, 16 KB data cache with a 32-byte block size and snooping logic enabled, an 8 KB local memory block for the vector accelerator segmented into two banks of 256 words by 128 bits. We selected a configuration implementing all the necessary extension instructions for MPEG-2 and H.264 acceleration. The accelerator includes a vector register file of 8 words, each of 128, bits in a 5R1W configuration. We synthesized the design flat on a modern physical synthesis tool and performed power planning and final routing in a commercial quality standard cell router. The resulting VLSI macrocell floorplan and layout of the dual-port register file configuration are shown in Figs. 15a and 15b respectively.

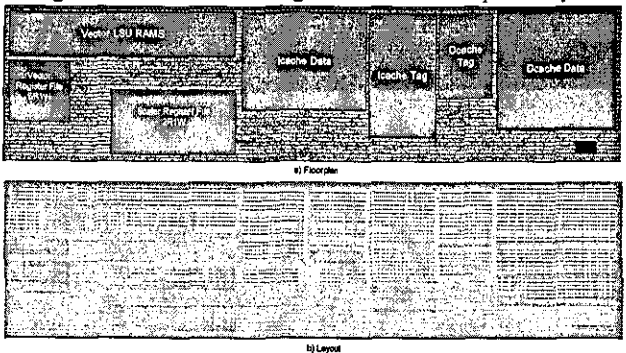


Figure 15: Floorplan and Layout of the Processor-Coprocessor architecture

Table 1 lists the implementation details of the placed and routed design.

Table 1: VLSI Macrocell Characteristics	
Instances/Macros	51156/26
Area (μm^2)	$3996 \times 996.3 = 3981693 \mu\text{m}^2$
Core Utilization	86.3%
Fmax (MHz)	194.1

IX. CONCLUSION AND FUTURE WORK

We described a configurable scalar-processor/vector-coprocessor microarchitecture for accelerating the existing and emerging video-coding standards. Architecture level experimentation identified a number of vector operations that significantly reduce the dynamic instruction count of MPEG-2, MPEG-4 and the ME function of a proprietary H264 implementation thus allowing for real-time video encoding on a wireless, portable device. Further work will focus on improving the achieved figures through the spatial re-arrangement of luminance data such that further data level parallelism can be exposed in the case of MPEG2. Most importantly, our investigations have shown that there exists a very significant amount of thread level parallelism in all workloads. We therefore plan to investigate cache-coherent multi-processor configurations of the proposed microarchitecture in an attempt to further reduce the complexity of the optimized encoders.

VIII. VLSI MACROCELL

We implemented a subset of the microarchitecture of Fig. 5

REFERENCES

- [1] G. Lawton, 'New Technologies Place Video in Your Hand', IEEE Computer, Vol. 34, No. 4, pp. 14-17, 2001
- [2] S. Vassiliadis, G. Kuzmanov, S. Wong, 'MPEG-4 and the New Multimedia Architectural Challenges', Proc. 15th International Conference on Systems for Automation of Engineering and Research (SAER-2001), pp. 24-31, Bulgaria, 2001.
- [3] 'Emerging H.26L Standard: Overview and TMS320C64x Digital Media Platform Implementation', White Paper, UB Video Inc., Vancouver, Canada, 2002
- [4] P. Orbaek, 'A real-time software video codec based on wavelets', Proc. Of Intl. Conf. On Communication Technology (IFIP), 2000
- [5] J. Streit, L. Hanzo, 'A Fractal Video Communicator', IEEE Vehicular Technology Conference (VTC), pp. 1030-1034, Stockholm, Sweden, 1994
- [6] TMS320C6000 CPU and Instruction Set Reference Guide', Document SPRU189F, Texas Instruments Incorporated, Houston, Texas 77251-1443, USA.
- [7] T. R. Halfhill, "Silicon Magic Breaks Out", Microprocessor Report, December 1st 2003
- [8] R. Bhargava, L. John, B. Evans, and R. Radhakrishnan, 'Evaluating MMX technology using DSP and multimedia applications' Proc. Of IEEE/ACM Sym. on Microarchitecture, pp. 37-46, December, 1998.
- [9] D. Talla, L. K. John, V. Lapinskii and B. L. Evans, 'Evaluating signal processing and multimedia applications on SIMD, VLIW and superscalar architectures', Proc. IEEE Int. Conf. on Computer Design, pp. 163-172, Sep. 2000
- [10] Equator Technologies MAP-CA', EDN.com, Online Edition available at www.e-insite.net/ednmag/, 2000
- [11] J. Kneip et. al., 'Applying and Implementing the MPEG-4 Multimedia Standard', IEEE Micro, Vol. 19, No. 6, pp. 64-74, 1999
- [12] 'Ax36 Family of Parallel Image and Video Digital Signal Processors DSP Chips', White Paper, Oxford Micro Devices, Inc, Monroe, CT 06468, USA, 2002
- [13] M. Long, 'Amphion Launches MPEG-4 Hardware-Accelerator Cores', e-inSITE, Online Edition available at <http://www.e-insite.net/esec/>, 2002
- [14] M. Long, 'Emblaze Semi Samples Multimedia Co-Processor for Handhelds', e-inSITE, Online Edition available at <http://www.e-insite.net/esec/>, 2003
- [15] 'VW2005 MPEG-1,2,-4 Audio/Video Encoder Chip', Product Brief, Vweb Corporation, San Jose, CA 9129, USA, 2002
- [16] 'MPEG-2 HDTV I&P Encoder', Product Specification, Duma Video, Inc., Portland, OR 97220, 2002
- [17] <http://www.mpeg.org>
- [18] <http://www.xvid.org>
- [19] K. Rijkse, 'H.263: Video Coding for Low-Bit-Rate Communication', IEEE Communications Magazine, pp. 42-45, December, 1996
- [20] [SULLIVAN] G.J. Sullivan, P. Topiwala, A. Luthra, "The H.264/AVC Advanced Video Coding standard: overview and introduction to the Fidelity Range Extensions", *SPIE conference on Applications of Digital Image Processing XXVII*, August 2004.
- [21] [JVT] Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, "Text of ISO/IEC 14496-10:2004 Advanced Video Coding Standard (second edition)", *ISO/IEC JTC1/SC29/WG11/N6359*, Munich, Germany, March 2004.
- [22] D. Alfonso, D. Bagni, L. Celetto, L. Pezzoni, "Detailed rate-distortion analysis of H.264 video coding standard and comparison to MPEG-2/4", in *Proceedings of Visual Communication and Image Processing (VCIP)* 2003, Lugano, Switzerland.
- [23] D. Alfonso, D. Bagni, L. Celetto, S. Milani, "Constant bit-rate control efficiency with fast motion estimation in H.264/AVC video coding standard", *Proceedings of the 12th European Signal Processing Conference (EUSIPCO)* 2004, Wien, Austria.
- [24] ISO/IEC JTC1/SC29/WG11, Test Model 5, April 1993
- [25] The Sparc Architecture Manual Version 8', <http://www.sparc.org>
- [26] 'The Leon-2 processor User's manual, XST edition, ver. 1.0.14', <http://www.gaisler.com>
- [27] D. Burger, T. Austin, 'Evaluating Future Microprocessors: The Simplescalar Tool Set', <http://www.simplescalar.com>
- [28] V. A. Chouliaras, J. L. Nunez-Yanez, S. Agha, 'Silicon Implementation of a Parametric Vector Datapath for real-time MPEG2 encoding', proceedings of the IASTED (SIP) 2004, Honolulu, Hawaii, USA
- [29] www.arm.com/armtech/AMBA_Spec?OpenDocument



Vassilios A. Chouliaras was born in Athens, Greece in 1969. He received a B.Sc. in Physics and Laser Science from Heriot-Watt University, Edinburgh in 1993 and an M.Sc. in VLSI Systems Engineering from UMIST in 1995. He worked as an ASIC design engineer for INTRACOM SA and as a senior R&D engineer/processor architect for ARC International. Currently, he is a lecturer in the Department of Electronic and Electrical Engineering at the University of Loughborough, UK where he is leading the research into embedded CPUs and SoC modeling. His research interests include superscalar and vector CPU microarchitecture, high-performance embedded CPU implementations, performance modeling, custom instruction set design and self-timed design.



José Luis Núñez is a lecturer in the department of Electronic Engineering at Bristol University. Prior to that he was a research fellow in the department of Electronic Engineering at Loughborough University where he worked since 1997. His current interests include the areas of lossless/lossy data compression, reconfigurable computing, FPGA-based design and high-speed data networks. He received his BS and MS degree in Electronics Engineering from Universidad de La Coruna (La Coruna, Spain) and Universidad Politécnica de Cataluña (Barcelona, Spain) respectively in 1993 and 1997. He received his PhD degree at Loughborough University (Loughborough, England) in 2001 working in the area of hardware architectures for high-speed lossless data compression



David J. Mulvaney has been a Senior Lecturer in the Department of Electronic and Electrical Engineering at Loughborough University since June 2001. He is currently managing research sponsored by a number of commercial and government bodies. His main research interests include novel real-time embedded machine learning techniques and electronic hardware solutions for real-time applications. Dr Mulvaney has carried out consultancy work for BP, Otis, Cadbury-Schweppes and GE Lighting, gives commercial training courses in real-time embedded C++ and has over 40 publications in professional journals and at international conferences.



Fabrizio S. ROVATI was born in Monza, Italy in 1971. He received electronic engineering degree at the Milan Polytechnic, Italy, in 1996. He joined STMicroelectronics Ltd., Bristol, UK (formerly INMOS Ltd.) where he contributed to the development of an MPEG-2 transport demultiplexer co-processor. He then joined STMicroelectronics' Advanced System Technologies in 1998 where he worked on the design of an MPEG-2 motion estimation co-processor and on MPEG video encoder's system architectures. He published four papers and holds seven patents and nine patent applications in digital video signal processing and architectures field. He is contract professor at Pavia Polytechnic University and gave several lectures at Milan Polytechnic University. His main interests are in digital video signal processing and related system-level and processors architectures.



Daniele Alfonso (M'04) was born in Alghero, Italy, in 1972. In 1998 he received a master degree in Electrical Engineering from the Turin Polytechnic, and then he joined STMicroelectronics, Advanced System Technology labs, working on image compression algorithms, jointly with the Italian National Research Council. Later, he focused on moving pictures encoding and transcoding (H.263, MPEG-2, MPEG-4, and H.264), low-power motion estimation, de-interlacing and frame-rate conversion. His main interests are algorithms and architectures for digital video applications and he holds several patents granted in Europe.

Paper PJ4: J. L. Nunez, V. A. Chouliaras, '*High Performance Arithmetic Coding VLSI Macro for the H264 Video Compression Standard*', IEEE Transactions on Consumer Electronics Vol. 51, Issue 1, Feb 2005, pg 144-151

High-performance Arithmetic Coding VLSI Macro for the H264 Video Compression Standard

J. L. Núñez, V. A. Chouliaras, Member, IEEE

Abstract — *this paper investigates the algorithmic complexity of arithmetic coding in the new H264 video coding standard and proposes a processor-coprocessor architecture to reduce it by more than an order of magnitude. The proposed coprocessor is based on an innovative algorithm known as the MZ-coder and maintains the original coding efficiency via a low-complexity, multiplication-free, non-stalling, fully pipelined architecture. The coprocessor achieves a constant throughput for both coding and decoding processes of 1 symbol per cycle and is designed to be attached to a controlling embedded RISC CPU whose instruction set has been extended with arithmetic coding instructions.*

Index Terms — arithmetic coding, H264, video coding, Golomb codes, renormalization, embedded systems, RISC CPU.

I. INTRODUCTION

The digitisation of visual information is nowadays common practice in large number of consumer products such as Personal Digital Assistants (PDA), digital cameras, palmtop computers and cellular phones as well as portable video game consoles and portable DVD players [1]. This exponential increase in the amount of digital visual information that must be stored, processed and then, transmitted efficiently has motivated a large body of research, both in industry as well as in academia, into advanced video coding techniques. New video coding standards such as the recent H264 video codec (also known as MPEG4 part 10) [2] deliver better quality and lower bit rates but at the expense of an almost exponential increase in the number of CPU cycles required per input frame of video data when compared to previous generation standards [3]. The introduction of advanced entropy coding within the H264 standard, via the pioneering use of context-based arithmetic coding [4], is one of the reasons behind the increase in the computational cost of the codec. The high-speed arithmetic coder (AC) coprocessor described in this paper has been designed to achieve a significant reduction of the AC computational cost of the H264 standard with modest hardware cost. This paper is organised as follows: Section II reviews hardware-based binary arithmetic coders. Section III presents the motivation for this work based on a study of the

computational costs of AC in the H264 video codec. Section IV presents our novel MZ coder and evaluates its efficiency compared with other arithmetic coding implementations. Section V studies the applicability of the MZ coder to entropy coding within the context of the H264 video standard. Section VI presents the hardware architecture of the MZ codec core and section VII describes the required ISA extensions. Section VIII presents the implementation results when targeting high performance FPGA and ASIC technologies. Finally, section IX summarizes our findings, and concludes this work.

II. HARDWARE-BASED BINARY ARITHMETIC CODERS

The IBM Q-coder [5] and the QM-coder [6] are the best known examples of hardware-based binary arithmetic coders. These devices use the *renormalization approximation* introduced by Rissanen in [7] to avoid the complex multiplications and divisions with the main difference across them being the complexity of the model which is formed by 128 contexts in the Q-coder and 1024 contexts in the QM-coder respectively. VLSI implementations of both the Q-coder and QM-coder are reported in [6]. Both hardware algorithms clocked at 75 MHz with a throughput of around 64 Mbits/second. The device was implemented in 0.35 μ m standard cell technology from IBM (CMOS 5S). The adaptive binary arithmetic coding device presented in [8] replaces the division operation by storing the probability values in a lookup table and uses the coder state as a pointer to a particular probability in that table. Similarly to the QM-coder, 1024 contexts are used each of them with its own probability state. Multiplications on the other hand are done explicitly using an 8x8 parallel multiplier. The VLSI implementation was carried out on TSMC's 0.8 μ m standard cell CMOS technology and the chip achieved a maximum frequency of 25 MHz. This device needs 8 clock cycles to complete the probability estimation and arithmetic operation phases plus a variable number of renormalization cycles. Renormalization typically is done in a single clock cycle but up to 7 clock cycles may be required. The resulting symbol throughput is approximately 3 Mbits/second. An improved version of that chip is presented in [9] in which a dynamic pipeline architecture is used to deliver 6 Mbits/second throughput at the same clock frequency and technology as the original design but with approximately 30% area overhead.

III. ANALYSIS OF AC IN THE H264 VIDEO CODEC

The H264 video coding standard is a state-of-the-art algorithm which delivers up to 50% reduction in bit-rates when compared to previous standards such as MPEG4 [10] or H263

J. L. Núñez is with the Department of Electronic and Electrical Engineering, University of Bristol, UK (e-mail: j.l.nunez-yanez@bristol.ac.uk)

V. A. Chouliaras is with the department of Electronic Engineering, University of Loughborough, UK (e-mail: v.a.chouliaras@lboro.ac.uk)

[11] at equivalent video quality settings. In order to achieve this the H264 uses a number of innovative techniques for extracting the redundancy present in the video stream. One such technique is the use of an arithmetic coding extension and context-based modeling known as CABAC [12]. The underlying architecture of the H264 standard is similar to that of previous standards such as the H263 and MPEG4, and consists of four major computational stages: Motion estimation and compensation (ME), discrete cosine transform (DCT), quantization (Q) and entropy coding (EC). These functions are applied to the data blocks resulting from dividing the image frame into variable size subframes. Finally, EC is applied after the binarization stage to the data produced by the ME and Q functions to further remove redundancy and reduce bit rates and it achieves this by using codes with fewer bits for the most probable parameters. In previous standards entropy coding was based in variable-length-codes (VLC) derived from the well-known Huffman codes due to their simplicity and ease of implementation. H264 can also use VLC codes but also permits the use of arithmetic coding as an improved alternative to VLC. In the approach used by the CABAC algorithm embedded in the H264 standard the probability state includes the probability information and multiplications are replaced by table look-up operations using the probability state and the two most-significant bits (MSB) of the range as pointers. This translates into a single table of 64 probability states times 4 ranges or otherwise, 256 8-bit values. Arithmetic is then carried out via simple additions and subtractions however, the renormalization loop is a sequential process with a cost of up to 6 iterations. The modelling stage in CABAC uses 2 additional tables for the adaptation process, to calculate the new probability state of the active context depending on whether a Most-Probable-Symbol (MPS) or a Least-Probable-Symbol (LPS) has just been coded.

To evaluate the complexity of AC in the H264 codec we selected 7 video sequences organized in 3 standard video formats namely QCIF, CIF, and SDTV. The video sequences and the chosen configuration of the H264 video codec are summarized in Table 1:

Video Format	Size (pixels)	Frame Count	Sequence Name	H264 Configuration
QCIF	176x144	60	Container, Foreman, News	ME search range = 16, ME full search, reference frames = 5, Hadamard transform ON, B frames ON, RD optimization ON
CIF	352x288	60	Paris, Tennis, Mobile	ON
SDTV	720x576	20	Calendar	ON

Table 1. Test Video Sequences and H264 configuration

Initial profiling of the algorithm was carried out in the SimpleScalar [13] processor simulation environment. Profiling data indicates that the complexity of AC during decoding is a fraction of the complexity during the coding process. Nevertheless, we chose to support both coding and decoding acceleration in the hardware coprocessor to offer a self-contained solution. These results are summarized in Table 2 for the QCIF format.

	Total Instruction Count per frame (Millions)	AC Instruction Count per frame (Millions)	AC calls per frame (Millions)	Instruction count per AC call
Coder	3,046	68.5	1.1	62
Decoder	23.5	0.2	0.0047	42

Table 2. AC Coding/Decoding Complexity in the QCIF Format

The number of calls to the AC routine increases substantially in higher quality settings and for larger video formats. This is illustrated in Figs. 1 and 2 which depict the number of AC calls per frame and the associated PSNR values, as a function of the quantization parameter QP. One AC call is needed for each bit of data (symbol) that must be coded. The AC costs range from 1 million calls per frame to approximately 60 millions calls per frame for a QP of 16, depending on the input video format. The high computation coding requirements are largely due to the *Rate Distortion Optimization* (RD) [14] in the H264 standard. In RD optimisation each macroblock is coded with different modes and the one that minimizes the rate-distortion curve is selected. In addition to the number of calls, algorithm profiling indicates that an average of 62 instructions for coding and 42 instructions for decoding are needed for each AC function call. When targeting an embedded, scalar, RISC CPU like the SPARC-compliant Leon2 [15] used in this work this translates approximately into 100 CPU clock cycles per AC call or bit for an average clocks-per-instruction (CPI) ratio of approximately 1.6 for the coding phase. We can therefore safely conclude that AC, in the context of advanced video coding, is a very compute-intensive operation and since traditional parallelizing techniques such as SIMD [16] extensions cannot accelerate this essentially sequential process, the introduction of dedicated hardware support in the form of a specialized coprocessor is a suitable solution.

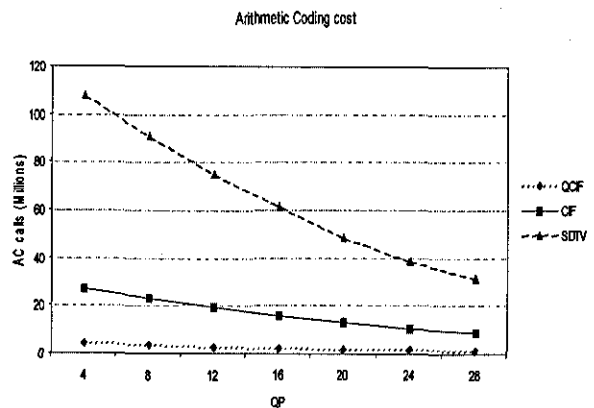


Fig. 1. H264 arithmetic coding complexity

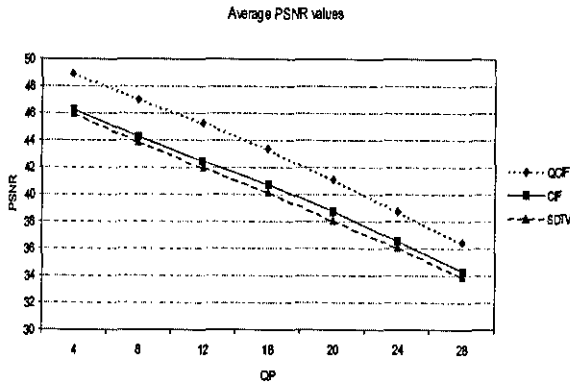


Fig. 2. H264 PSNR evaluation

IV. PROPOSED ARITHMETIC CODING ALGORITHM

The original Z-coder software algorithm was developed by ATT labs [17] as a generalization of the Golomb/Rice coder for lossless coding of bi-level images. Golomb/Rice coding is used to code a run of r consecutive occurrences of a MPS followed by a single occurrence of a LPS, using a parameter m to control how many MPS fit in one bit of code and also how many bits of code are required to code a LPS. The code has two components: the first component is r/m 1's, followed by a single 0, while the second component is $r \bmod m$, coded as an ordinary binary number with $\log_2(m)$ bits. Although easy to implement, the limitation of Golomb codes is that the chosen parameter m is only good for a single probability distribution however, a general compression system has to be able to deal with arbitrary sequences of events with different probabilities. The Z-coder aims to solve this limitation. Z-coding is the same as Golomb coding with the advantage that the parameter m can be changed for each symbol being coded. The extra complexity of the algorithm is small and more details can be found in the original paper [17]. Our work has focused on maintaining the simplicity of the Z-coding algorithm while increasing its suitability for hardware implementation and this is where our novelty lies. The resulting MZ-coder balances the complexity of coding the MPS and LPS, simplifies the precision of the arithmetic and handles special hardware borrow conditions while maintaining coding efficiency and achieving high performance via a fully pipelined micro-architecture. In order to validate the efficiency of the MZ-coder in a general compression environment a software implementation has been developed in which a sophisticated variable-order Markov-model [18] has been coupled to a selection of 3 arithmetic coders named the Lei coder, the Bmult coder and our own MZ coder. The Lei coder improves the coding efficiency of the Q-coder and a detailed description is available in [19]. The Bmult algorithm uses the standard method proposed in [18] with full precision integer multiplications. These two known arithmetic coders and the MZ-coder have been compared with the information content of

the Markov modeler measured by the equation $\text{symbol bits} = -\log_2(\text{symbol probability})$ using floating point arithmetic. This equation bounds the theoretical compression for the given model. Our experimentation is based on two standard data sets commonly used in the literature: the *Calgary* and the *Canterbury* [20] data sets. Figs. 3 and 4 show the percentual compression degradation (Y axis) as a function of the block size (X axis). The best performer is, as expected, the Bmult algorithm using full precision integer multiplications. The two multiplication-free coders perform similarly with a maximum degradation of around 1% although the MZ coder outperforms the Lei coder in all block sizes. Additionally, the MZ coder performs very well for small block sizes outperforming the information content of the model given by the floating point arithmetic. The reason is that the MZ algorithm has been designed to predict symbols with a slightly higher level of confidence than that obtained from the probability data provided by the model. Extensive simulation has shown that slight over-predictions are particularly beneficial for small block sizes where the limited amount of data available prevents model construction from entering a stable state.

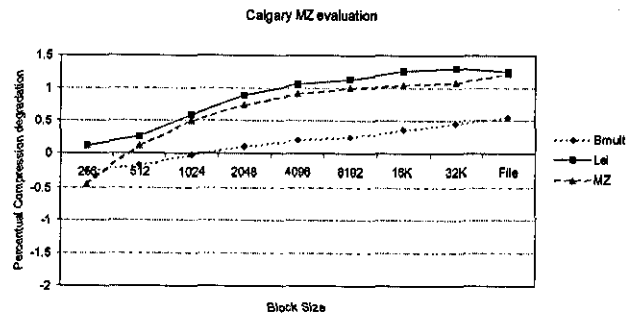


Fig. 3. MZ arithmetic coding efficiency in the Calgary corpus

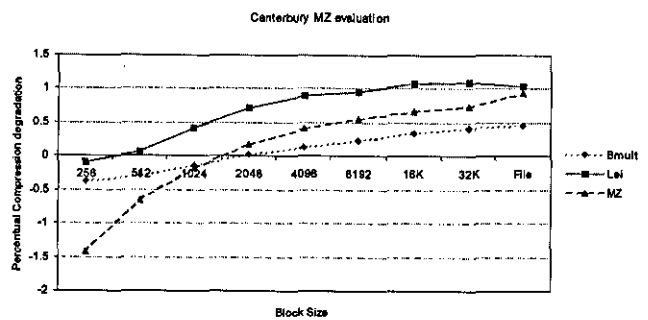


Fig. 4. MZ arithmetic coding efficiency in the Canterbury corpus

Apart from offering good coding efficiency, one of the main attractive points of the MZ-coder is its fast renormalization. The original AC algorithm present in CABAC uses a variable cycle (from 0 to a maximum of 6 cycles) renormalization stage to keep the state variables in the required range. This variable renormalization latency is due to the inner dependencies of the state variables and the renormalization loop. Fig. 5 illustrates

that the costs of multiple-cycle renormalization account for a performance degradation of around 15%.

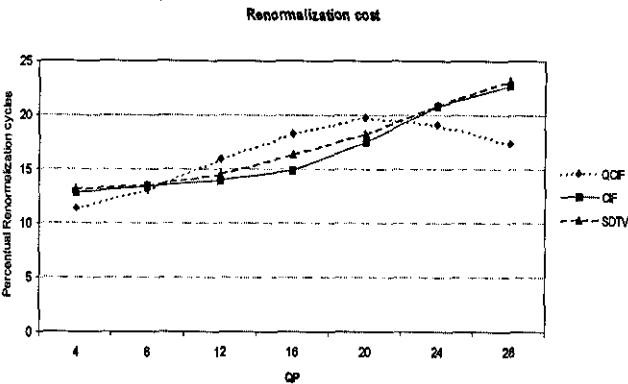


Fig. 5. Renormalization costs in CABAC

On the other hand, the renormalization process in the MZ-coder does not include internal dependencies. As a result, it can be readily accomplished with a shift left operation. This is illustrated in the pseudocode of Fig. 6 which also shows the internal dependencies of *low* inside the while loop in the CABAC case. This MZ-coder feature guarantees a data-independent throughput of 1 symbol per clock cycle and simplifies the control data path.

```
pLPS = table64x6(state);
rLPS = table256x8 (state,range);
Z = range + pLPS;
range = range - rLPS;
if (Z >= HALF)
    Z=QUARTER + Z >>1;
if (symbol == MPS)
    {
        low+=range;
        range = rLPS;
        range = Z;
        if (range >= HALF)
        {
            output 1 bit;
            range <=1;
            subend <=1;
        }
        /*renormalization loop*/
        while( range < QUARTER)
        {
            if(low >= HALF)
            {
                Output bit;
                low -= HALF;
            }
            else
            {
                if (low < QUARTER)
                    Output bit;
                else
                    bits to follow++;
                low=QUARTER;
            }
            low <= 1;
            range <=1;
        }
    }
else
{
    Z = FULL - Z;
    subend +=Z;
    range += Z;
    shift_bits = shift(range);
    output shift_bits bits;
    range <= shift_bits;
    subend <= shift_bits;
}
```

Fig. 6. CABAC & MZ pseudocode description

V. VIDEO CODING EFFICIENCY OF THE MZ ALGORITHM

The MZ algorithm has been incorporated into the JM 7.3 H264 reference software [21] and its coding efficiency measured using the video sequences of Table 1.

Figs. 6 and 7 depict the coding efficiency of the proposed MZ coder and the VLC coder versus the CABAC algorithm. Fig. 6 shows that the performance of CABAC and the MZ coder are virtually undistinguishable. On the other hand, the simple VLC codes increase the bit rates by around 8% for these video sequences with the effect being more noticeable for the large SDTV format.

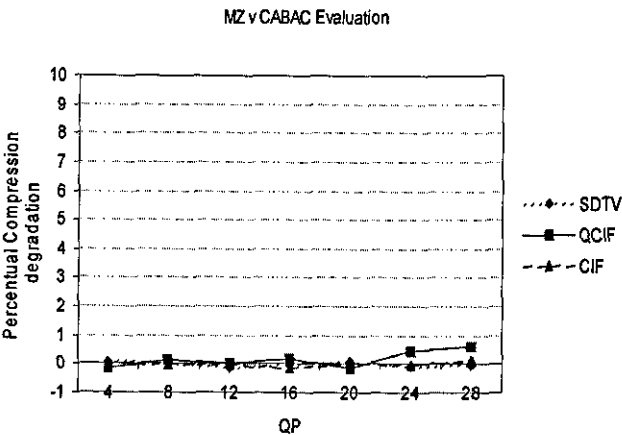


Figure 6. MZ coding efficiency

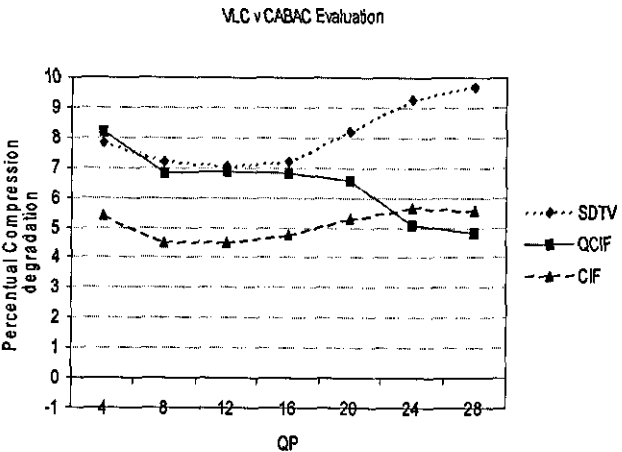


Figure 7. VLC coding efficiency

These results have been verified by decoding the resulting bit files using the corresponding entropy decoders for each of the 3 options tested (VLC, MZ, CABAC). They validate the MZ algorithm as delivering the same level of performance as the original CABAC. The unique advantage of the MZ is therefore its single cycle renormalization capability and its efficient hardware micro-architecture which results in high performance VLSI implementation. The microarchitecture is presented in the following sections.

VI. ARCHITECTURE AND VLSI IMPLEMENTATION

Fig. 8 shows the hardware architecture of the arithmetic coding coprocessor.

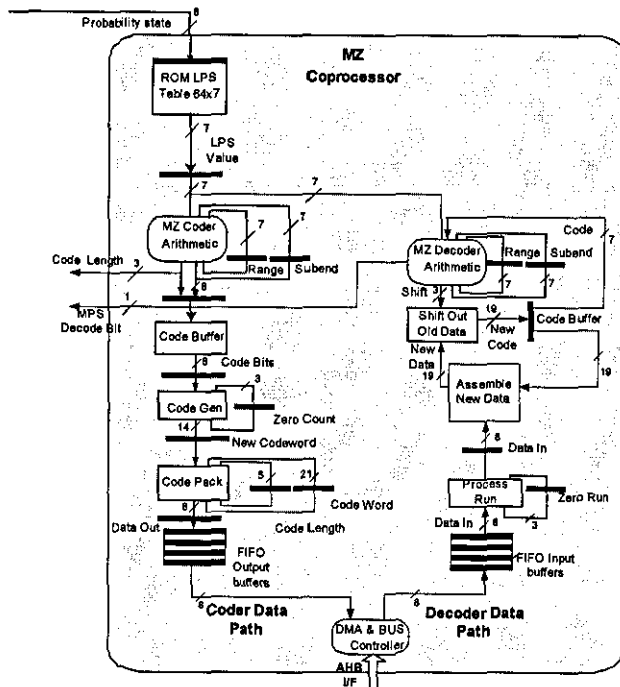


Figure 8. Coprocessor architecture

The coprocessor has been coupled to a SPARC V8 [22] compliant embedded CPU [15] which includes a standard, 5-stage RISC pipeline. The Leon2 processor was selected for this work due to its open-source nature which makes the integration of the coprocessor pipeline in the Leon2 data path easier due to having full access to the RTL source code. The following sections describe the main modules of the MZ-coder.

A. Arithmetic Coding Coprocessor Description

1) MZ coder arithmetic

The hardware implementation reduces the arithmetic precision to 8 bits and the precision of the *subend* and *range* registers to 7 bits from the original 17 bits and 16 bits in the Z-coder [17] software algorithm respectively. This precision is sufficient to handle the minimum symbol frequency of 1/128 as fixed by the LPS table, without affecting coding efficiency. The renormalization is done in parallel for range and subend and in the same pipeline cycle as the rest of the MZ arithmetic. The number of bits that must be added to the output code depends on the amount of renormalization needed in the range value so that the range is kept between "0000000" and "1000000". Shifting must be done until the MSB of the range value is 0. The shift value ranges from 0 when no shifting is required to 7 when the input range is "1111111" and 7 shift operations are

required to obtain "0000000". The code bits output from the MZ arithmetic stage are buffered in the code buffer stage.

2) Code buffer

The code buffer stage is required to control possible borrow bits originating in the previous stage that could affect the value of the bits contained in the code buffer. A total of 8 bits are buffered in this stage. A number of bits, as defined by the shift value, must be inserted at the least-significant bit in the code buffer. The result from doing the MZ arithmetic means that an overflow is possible in the subend register. As long as the value stored in the buffer is different from 0, the borrow will be stopped in the code buffer register. If the value of the buffer is 0 then the borrow propagates out into bits that have already been sent to the code generator stage (discussed next) and the current output is formed by as many bits set to 1 as specified by the shift signal since borrow propagation, in the code buffer, will swap all the bits from 0 to 1. The code generator stage handles possible borrows originating in the code buffer by not outputting bits until a bit set to 1 has been received from the code buffer stage. The bit set to 1 will behave as a barrier for the possible borrows being propagated out of the code buffer.

3) Code generator

The code generator takes the 0 to 7 bits produced by the code buffer and the zero run count to build a code of up to 14 bits. The *zero run* register counts the number of consecutive 0 bits in the input. These bits are the equivalent of the *bits_to_follow* variable used by software arithmetic coders [4]. The output is formed as a bit set to 1 plus zero run bits set to 0 when the first bit set to 1 from the code buffer is received after a run of consecutive 0's. Output is then possible since the bit set to 1 will block any possible borrows originating in any following coding events. In software the *bits_to_follow* counter is a simple integer variable but in hardware this could leave an undefined and potentially unlimited number of bits in the coder pending to be output. This is undesirable from a latency and complexity point of view so instead of using a large 32-bit register, a 3-bit counter is utilized to keep track of the zero run count. This mechanism means that only a maximum of 7 bits could be left in the coder pending to be output. The maximum length codeword that the bit packer should be able to handle is therefore given by:

$$\begin{aligned} \text{Max length codeword} &= 7 \text{ new bits} + 7 \text{ bits pending} = \\ &= 14 \text{ bits.} \end{aligned}$$

It is possible that more than 7 bits set to 0 are received in which case the zero run counter will overflow. To avoid this situation the hardware emits the pending 7 bits set to 0 preceded by a bit set to 1 in a speculative manner. The first bit of the next output codeword will indicate if the bits emitted speculatively must be inverted. The decoder will extract this bit from the data stream, negate it and subtract the result from the previous code, effectively transforming any code bit sequence of "10000000" to "01111111". This process adjusts the code to the correct value and performs a similar function to

the stuffing bit suggested by IBM in their Q-coder [5]. The extra bit is part of the next codeword and will have the value 0 if and only if a borrow bit originated in the code buffer stage. Potentially a run of 7 consecutive 0s could be followed by another run of 7 consecutive 0s overflowing the zero run count again. This is not a problem and the hardware will emit codewords as normal. The potential long borrow will not cause the decoder to fail because all the coding events that were coded previous to the event that produced the borrow can be decoded without any borrow propagation. The fundamental requirement to guarantee correct decoding is that the borrow must be propagated in the decoder before the decoder tries to decode the bit that produced that borrow in the first place.

4) Code packer

The variable number of bits produced by the code generator is finally pipelined to the code packer whose function is to pack the variable length codewords into fixed-length 8-bit codewords, ready to be output. Since up to 7 bits can be left inside the code packer without generating any output and up to 14 bits can be forwarded by the code generator stage in every cycle, the width of the packer register has to be at least 21 bits to be able to store all the data bits in this particular case.

B. Arithmetic Decoding Coprocessor Description

1). Process run

The module to process the zero runs checks if 7 consecutive bits are set to zero with the help of the zero run register. If this condition is detected, the next bit corresponds to an extra bit added by the coder. This bit is removed from the coded data stream and a *borrow_propagate* signal is forwarded to the next pipeline stage to adjust the rest of the codeword bits accordingly, before they are used by the decoder. If the bit is set to 1 no borrow propagation is needed but if the bit is set to 0 a borrow propagation must take place that will be stopped by the first bit set to 1 in the code buffer register.

2) Assemble new data and Shift out old data

This block buffers the codeword bits before they are required by the MZ-decoder arithmetic. To increase the amount of parallelism between the MZ decoder arithmetic and the assembly-shift operation, data is concatenated in the *assemble_new* data module before the arithmetic logic has determined how many bits must be disregarded. Once this value is known, old data is shifted out and the codeword is rebuilt using the arithmetic adjusted codeword and new data in the *shift_out_old_data* module. The codeword is then registered in the code buffer, ready to start a new decoding operation. Since assembling of new data is done without knowing how many bits are going to be disregarded by the decoding arithmetic, enough bits must be present so that, in the case that no data is assembled but maximum bits are disregarded, enough valid bits remain for the next decoding cycle. It is also critical to propagate the borrow signal as far as the first bit set to 1. At least 6 bits of codeword must always be valid. If a decoding operation can consume up to 6 bits and at least 6 bits must remain valid for the next decoding operation

at least 12 bits must be valid at the start of the cycle. This means that data must be added when less than 12 valid bits remain in the code buffer register. The code buffer register must therefore be at least 19 bits wide to be able to store the total of 8 new bits plus 11 bits of codeword.

3). MZ decoder arithmetic

The decoding arithmetic follows that presented in the Z-coder paper [18] with the benefit of using only 7-bit precision and having balanced MPS/LPS branches, similar to the proposed coding hardware. The arithmetic circuits have been designed to maximise throughput by performing as many operations in parallel as possible.

Finally, the DMA/AHB bus controller moves data between the internal coprocessor FIFOs and main memory.

VII. ISA EXTENSIONS

A total of 5 instructions have been added to the SPARC V8 ISA to support the coprocessor. The *MZ_code_mps* and *MZ_code_lps* instructions advance the MZ pipeline and are used each time the main processor enters the AC routine. The data transferred to the MZ module when any of these two instructions is executed is the 6-bit probability state. The MZ arithmetic corresponding to an MPS or LPS coding event is then executed and the results are forwarded to the next pipeline stage (code buffer). The MZ state stored in the registers *range* and *subend* is also updated. The data path from the MZ arithmetic to the execution stage in the Leon processor returns the number of bits needed by the executed coding instruction. The video codec software uses this information to calculate the current coding bit costs. The rate distortion optimisation accepts or rejects a sequence of coding events depending on the value of this cost and the current PSNR value. This means that two extra instructions are required to accept or reject previously executed coding events: *MZ_comit* and *MZ_reset*. Additionally and not shown in the figure but implied, there is a set of equivalent MZ state registers corresponding to the hidden state. These registers are updated by the *MZ_comit* instruction and are used to update the visible state by the *MZ_reset* instruction. Therefore, the purpose of the *MZ_comit* and *MZ_reset* instructions are to accept or reject previously coding events by updating the coprocessor state registers. The decoder requires another instruction extension called *MZ_decode_s*. Once the decoding process starts, the coprocessor state machine (not shown) fills up the code buffer register independently of the code running on the main CPU. Once a decode instruction is received, some of these bits are used to generate a decode MPS/LPS signal that indicates if a most probable symbol or a least probable symbol has been decoded. The software running on the main CPU interprets this signal as a bit set to 1 or a bit set to 0 depending on which symbol (0 or 1) is the most probable symbol. A valid signal indicates to the main CPU if the code buffer contained enough bits for the instruction to complete. Otherwise, the main CPU must reset the state of the decoder engine using *MZ_reset* and execute again the decode instruction. Finally, the state registers are pipelined at each level and move with the data

path pipeline with the rest of the codeword data. This is necessary to handle possible exceptions originating in the main CPU data path that would cause the main pipeline to restart and the same instruction to be executed more than once. A restart signal originating in the exception logic unit of the Leon2 CPU will load the pipeline state information into the corresponding state registers in the MZ coprocessor should a software exception happens in the main processor.

VIII. IMPLEMENTATION

To verify the functionality and performance of the AC coprocessor we have integrated the core in a SoPC platform implemented using an Altera APEX20KE PCI development board. The main components of the SoPC platform are illustrated in Fig. 9.

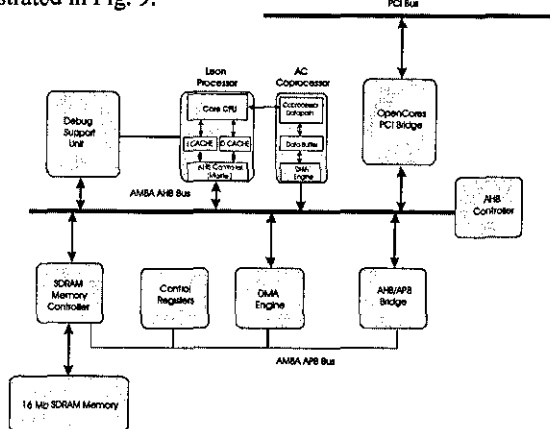


Figure 9. SoPC Platform

The AMBA AHB subsystem incorporates a total of 5 masters (Debug Support Unit, Leon2 Processor, AC coprocessor, DMA Engine, PCI Bridge Interface) and 2 slaves (memory controller and AHA/APB Bridge). The Control registers module, instantiated as a slave in the AMBA APB bus, controls the execution of the H264 binary on the FPGA board. There are a total of 5 extra registers added to the standard Leon system for control purposes. The interrupt register is hardwired to the open drain INTAN signal on the PCI bus. When one of the bits in the interrupt register is set to zero the INTAN signal goes low. It is the responsibility of the application driver running on the host computer to remove this interrupt by writing 0xFFFFFFFF to the interrupt register. The debug support unit can be used to help debugging an application running on target hardware. The MZ coprocessor can clock up to 50 MHz in this technology but the Leon2 processor and the OpenCores PCI Bridge are limited to 33 MHz. The complexity and performance details of the FPGA implementation are shown in Table 3. The AC coprocessor reduces the complexity of arithmetic coding by more than an order of magnitude in this configuration. The chosen Silicon technology for the VLSI macro was the UMC 0.13 μ m, 8-copper process. The design was originally synthesized in Synplify ASIC and then, read into the Synopsys Design Compiler for further logical netlist optimization. It was

then read into Synopsys Physical Compiler tool and optimized for Minimum Physical Constraints (MPC). The MPC (placed) netlist was then run through Place and Route on the Cadence Encounter platform to verify that the design was indeed routable.

Component	Logic Cells	Memory bits	Performance
Leon CPU	7724	94332	33 MHz
DSU	1355	16384	
PCI Bridge	3730	32768	
Memory Controller	733	8196	
DMA Engine	291	0	
AHB Controller	565	0	
AHB/APB Bridge	317	0	
Control Registers	400	0	
AC Coprocessor	1460	448	50 MHz
Overall SoPC	16575 (43% of APEX20K1000E Altera FPGA)	151828 (46% of APEX20K1000E Altera FPGA)	33 MHz

Table 3. SoPC complexity and performance details

Once the routability aspect of the design was achieved, the original logical netlist was read into Physical Compiler once more, but now with real physical constraints applied. These constraints specified the utilization factor, aspect ratio and die size (derived from the previous MPC run), power ring dimensions, power trunks width and number, pin (port) location and finally, the power straps characteristics. It was re-optimized and passed to SoC Encounter for the final Place and Route run. The maximum operating frequency was 330 MHz worst-case (throughput of 330 MSymbols/second) and the complexity of both the coder and decoder is 5600 standard cells. Fig. 10 depicts the final placement and layout of the arithmetic coding/decoding coprocessor.

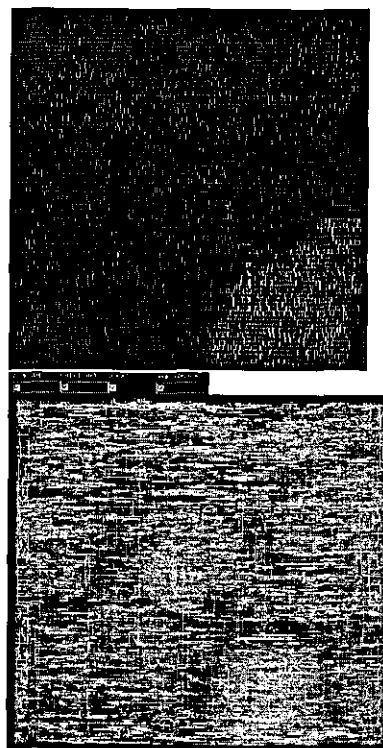


Figure 10. Coprocessor Placement and Layout

IX. CONCLUSIONS

This paper presented an innovative hardware architecture for arithmetic coding based on the simple Golomb codes that enables a data-independent throughput of 1 symbol per clock cycle without affecting coding efficiency. The MZ-coder has been applied to the problem of accelerating the compute-intensive entropy coding functions in the state of the art H264 video coding standard and shown to deliver equivalent bit rates while eliminating the need for multiple renormalization cycles. The hardware has been verified using low-cost FPGA technology and shown to have modest requirements in terms of silicon area while achieving good results in terms of clock rate. The SoPC platform utilizes the open-source Leon2 processor with the proposed accelerator and shown to reduce the complexity of AC by more than an order of magnitude. Subsequently, a VLSI implementation was carried out in a high performance 0.13 μm silicon process and the resulting macrocell achieved a throughput of 330 Msymbols/second. The H264 video coding standard is expected to be the enabling technology in the near future for personal multimedia communications. Major efforts are currently active within industry and academia to accelerate the compute-intensive motion estimation, transform and quantization functions through developing fast algorithms and exploiting the available data level parallelism. Entropy coding, based on arithmetic coding, is mainly a sequential process, not well suited to this kind of optimization. Its acceleration with the proposed hardware architecture could play a major role in bringing real-time H264 video coding within the grasp of low-power embedded devices.

References

- [1] G. Lawton, "New Technologies Place Video in Your Hand", IEEE Computer, Vol. 34, No. 4, pp. 14-17, 2001.
- [2] G. Bjontegaard, "H.26L Test Model Long Term Number 4 (TML 4) draft0", ITU-TSG16/Q.6 Q15-J-72, June 2000.
- [3] V. A. Chouliaras, J. L. Núñez "A Multi Standard Video Coding Accelerator based on a Vector Architecture", to appear in IEEE International Conference on Consumer Electronics, Las Vegas, January, 2005.
- [4] G. Langdon, "An Introduction to Arithmetic Coding", IBM J. Res. Develop, Vol. 28, No. 2, pp. 135-149, March 1984.
- [5] W.B. Pennebaker et al, "An overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder" IBM J. Res. Develop, Vol. 32, No. 6, pp. 717-725, November 1988.
- [6] M. J. Slattery, J. L. Mitchell, "The Qx-coder", IBM Journal of Research and Development, Vol. 42, No. 6, pp. 767-784, 1998.
- [7] J. Rissanen, K. Mohiuddin, "A Multiplication-free Multialphabet Arithmetic Coder", IEEE Transactions on Communications, Vol. 37, pp. 93-98, 1989.
- [8] S. Kuang, J. Jou, Y. Chen, "The Design of an Adaptive On-Line Binary Arithmetic- Coding Chip", IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 45, No. 7, pp 693-706, July 1998
- [9] S.R.Kuang et al., "Dynamic pipeline design of an adaptive binary arithmetic coder", IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 48, No. 6, pp. 813 -825, Sep 2001.
- [10] J. Kneip et. al., "Applying and Implementing the MPEG-4 Multimedia Standard", IEEE Micro, Vol. 19, No. 6, pp. 64-74, 1999.
- [11] K. Rijkse, "H.263: Video Coding for Low-Bit-Rate Communication", IEEE Communications Magazine, pp. 42-45, December, 1996.
- [12] D. Marpe, H. Schwartz, T. Wiegand, "Context-Based Adaptive Arithmetic Coding in the H.264 Video Compression Standard", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 620-636, 2003.
- [13] Information available at www.simplescalar.org
- [14] T. Stockhammer, D. Kontopodis, T. Wiegand, "Rate-distortion optimization for JVT/H.26L video coding in packet loss environment", Proc. of 2002 Int. PacketvideoWorkshop, Pittsburgh, USA, 2002.
- [15] Information available at www.gaisler.com
- [16] D. Talla, L. K. John, V. Lapinskii and B. L. Evans, "Evaluating signal processing and multimedia applications on SIMD, VLIW and superscalar architectures", Proc. IEEE Int. Conf. on Computer Design, pp. 163-172, Sep. 2000.
- [17] L. Bottou, P. G. Howard, Y. Bengio, "The Z-coder adaptive binary coder", In Proceedings of the Data Compression Conference, pages 13-22, March 1998.
- [18] J. Cleary, I. Witten, "Data Compression Using Adaptive Coding and Partial String Matching", IEEE Transactions on Communications, Vol. 32, No. 4, pp. 396-402, 1984
- [19] S. M. Lei, "Efficient Multiplication-Free Arithmetic Codes", IEEE Transactions on Communications", Vol. 43, No. 12, pp. 2950-2958, 1995
- [20] R. Arnold, T. Bell, "A Corpus for the Evaluation of Lossless Compression Algorithms", Data Compression Conference, pp. 201-210, 1997.
- [21] Sources available at <http://iphone.hhi.de/suehring/ttml/>
- [22] Information available at www.sparc.com/standards/V8.pdf



José Luis Núñez is a lecturer in the department of Electronic Engineering at Bristol University. Prior to that he was a research fellow in the department of Electronic Engineering at Loughborough University where he worked since 1997. His current interests include the areas of lossless/lossy data compression, reconfigurable computing, FPGA-based design and high-speed data networks. He received his BS and MS degree in Electronics Engineering from Universidad de La Coruna (La Coruna, Spain) and Universidad Politécnica de Cataluña (Barcelona, Spain) respectively in 1993 and 1997. He received his PhD degree at Loughborough University (Loughborough, England) in 2001 working in the area of hardware architectures for high-speed lossless data compression



Vassilios A. Chouliaras was born in Athens, Greece in 1969. He received a B.Sc. in Physics and Laser Science from Heriot-Watt University, Edinburgh in 1993 and an M.Sc. in VLSI Systems Engineering from UMIST in 1995. He worked as an ASIC design engineer for INTRACOM SA and as a senior R&D engineer/processor architect for ARC International. Currently, he is a lecturer in the Department of Electronic and Electrical Engineering at the University of Loughborough, UK. His research interests include superscalar and vector CPU microarchitecture, high-performance embedded CPU implementations, performance modeling, custom instruction set design and self-timed design.

Paper PJ5: Grecos, C., Saparon, A. and **Chouliaras, V.**, *'Three novel low complexity scanning orders for MPEG-2 full search motion estimation'*, Real Time Imaging, 10, February 2004, pp 53-65

Three novel low complexity scanning orders for MPEG-2 full search motion estimation

Christos Grecos*, Azilah Saparon, Vassilis Chouliaras

Department of Electronic/Electrical Engineering, Loughborough University, Leicestershire LE11 3TU, UK

Abstract

Complexity localisation in the reference frame is the key process for the derivation of efficient scanning orders for motion estimation. The more localised the complexity is, the more computationally efficient scanning orders can be derived for reduced cost motion estimation algorithms. However, this process entails serious pre-processing overhead which may render it unsuitable for real time video coding systems. In this paper, we propose three low complexity scanning orders of similar performance that are very competitive in terms of the operation count ratio metric with respect to the MPEG-2 raster scan order, show improvements of 7.14% on the average with respect to the number of examined macroblock rows metric and they also show an increase in the speed-up ratio of 0.12 on the average as compared to the standard. As compared to other work in the literature, the proposed scanning orders require one fourth of the operation count ratio and show an increase in the speed-up ratio of 45 times on the average.

© 2004 Elsevier Ltd. All rights reserved.

1. Introduction

The research efforts for the motion estimation problem in video coding can be classified in two categories according to the quality-complexity trade-off. The first category consists of the Full search methods [1] which examine all locations inside a given search window in the reference frame. These algorithms find the best match of the macroblock to be encoded in a window of a given size at the cost of high computational complexity. The second category consists of methods that trade off quality for reduced complexity, thus examining only a subset of locations inside a given search window. Such methods include the 2-d logarithmic search [3], the cross search [5], the Three Step Search (3SS) [2], the Four Step Search (4SS) [6], the gradient descent search [7], the diamond search [8] and a whole range of zonal searches [9]. To reduce the cost of the Full Search methods, a variety of schemes have been proposed in the literature such as the partial distortion elimination technique (PDE) [10] and its variations [15–17], successive elimination algorithms (SEA)

[18–24], fast 2-d finite impulse response filters [25], vertical–horizontal and massive projection techniques for candidate and reference blocks [26–27] etc. All these schemes examine only a subset of the pixels for a given search position in the reference frame and/or for the macroblock to be encoded, while still finding the best match in statistical terms. The work in this paper falls in this category of schemes.

An appealing possibility in the search methods without quality degradation is the tailoring of the scanning order for the computation of the sum of absolute differences (SAD) metric according to the direction of the local motion field in the reference frame. The idea is that the sooner the maximal changes of the local motion field are identified in the process of SAD computation between the macroblock to be encoded and a candidate predictor in the reference frame, the faster this predictor can be rejected if it has exceeded the minimum SAD found so far. Early search termination (or early jump out (EJO) in MPEG-2 terminology) obviously implies computational savings since only a subset of the pixels for the macroblock to be encoded and its potential best match in the reference frame need to be examined. The search will then continue from the next search position in the reference frame inside the given search window. This early bailing out mechanism essentially classifies these scanning orders as variable complexity algorithms (VCAs) since complexity of the

*Corresponding author. Tel.: +44-1509-227077; fax: +44-1509 227014.

E-mail addresses: c.grecos@lboro.ac.uk (C. Grecos), a.saparon@lboro.ac.uk (A. Saparon), v.a.chouliaras@lboro.ac.uk (V. Chouliaras).

motion estimation clearly depends on the properties of the input source, thus specific scanning orders that are tailored to these properties can indeed achieve computational scalability [14]. It has to be noted that the scanning orders we are dealing with refer to *motion estimation* rather than to the *encoding of DCT coefficients* despite the fact that the latter have been very popular in the literature (zig-zag scanning, alternate scanning etc. [4]). Due to simplicity in implementation, the MPEG-2 standard [1] in TM-5 chose to examine EJO points at the end of every macroblock row and it also chose a *fixed raster-scan order for the SAD computation of the pixels involved*. This fixed raster-scan order, which accumulates SAD in the two macroblocks from top to bottom and from left to right, can be very inflexible and cause unnecessary computational overhead due to the following reason: No attempt is made in the standard to identify content-wise the areas in the reference macroblock that will result in the biggest computational reduction if examined first, during the motion estimation process. This is in contrast with computationally intensive but theoretically proven results by Tao and Orchard [30] and later work by Kim [11–12], which clearly demonstrate that the early identification of regions with highest gradients (i.e. activity) in the reference macroblock will produce the maximal computational savings. Due to these theoretical and practical results [11,12,30], the *fast* identification of these high activity areas in the reference macroblock, either through blind estimation or through very low overhead classification that justifies its use is imperative.

The idea behind the first two blind estimation scanning orders we propose, is that by spiralling inwards in the reference macroblock, we are able to locate more pixels with higher gradients faster as compared to the raster scan order and thus potentially save computations. A simple predictor of the maximal change of the motion field, in combination with inward movement in the reference macroblock during the SAD calculations is the basis of the third scanning order proposed and it also achieves faster location of high gradient pixels. The following figure illustrates the benefits of the proposed schemes.

In Fig. 1, it is clear that the dissimilar areas (represented as white patches in the reference frame) of candidate macroblock predictors, can be faster identified and potentially eliminated by different scanning orders than the one used by the standard. In the cases depicted in the figure, a scanning order with EJO points at the ends of rows/columns from the right-most column to the left-most, from the left-most column to the right-most and from the bottom-most row to the top-most would be ideal for fast rejection of the first three candidate predictors. For the predictor represented by the free form closed contour inside the

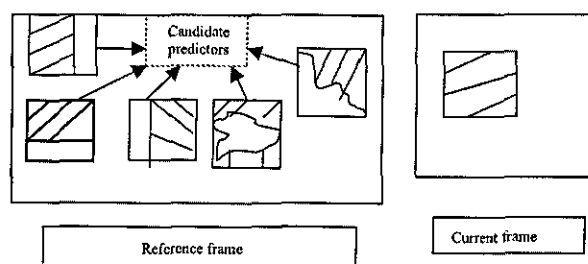


Fig. 1. Similarity between macroblocks of the current and reference frames.

reference macroblock, the highest gradient pixels will occur just on the borders of the contour where the texture changes. It can be easily seen that scanning based on squares of decreasing sizes as we are moving inward during SAD calculation in the reference macroblock, will be able to locate more boundary contour pixels faster since it examines four directions simultaneously as compared to the single scan direction used by the standard for the same number of operations per macroblock. This is also true for the candidate predictor represented by the free form open contour inside the reference macroblock, as shown in Fig. 1.

The design of scanning orders for Full search motion estimation algorithms can be subdivided in zero overhead and in limited pre-processing schemes. In zero overhead schemes, the order of SAD computation between pixels in the current and reference macroblocks is usually fixed and the choice of the order tries to compensate for the blindness of the estimation. Relevant work can be found in [12]. In limited pre-processing schemes, the maximal changes in the local motion field are identified through some kind of pre-processing such as gradient estimation, gradient sorting etc. [11]. It has to be noted that the gains of the limited pre-processing methods through accurate identification of the maximal changes in the local motion field *have to be weighted against the time that such pre-processing needs* for real time motion estimation schemes. This is the reason that we present in Section 4 both computational savings in terms of the average number of rows examined per macroblock, computational savings in terms of the operations required but also computational savings in terms of the actual run times for the different schemes, since the latter two metrics consider pre-processing overhead as well.

In this paper, we propose three novel scanning orders for motion estimation. The first two belong to the category of zero overhead schemes since they attempt to eliminate unsuitable predictors in the reference frame as fast as possible, based on joint exploitation of horizontal and vertical SAD information. The third scanning order proposed belongs to the category of limited pre-processing schemes, since it utilises SAD information *only of the boundary macroblock rows and columns* for

determining the scan direction. The paper is organised as follows: Section 2 describes the proposed scanning orders for motion estimation, Section 3 describes complexity comparisons of the proposed schemes as compared to other work in the literature and Section 4 contains comprehensive experiments that show the merits of the proposed scanning orders versus the raster-scan order used in MPEG-2 but also as compared to other works in the literature. Section 4 ends by drawing conclusions from our work.

2. Proposed scanning orders

2.1. Spiralling inward scanning order

This scanning order is based on the idea that the SAD value between pixels located on corresponding positions on the sides of squares of decreasing size inside the current and reference macroblocks, may be used to reject candidate predictors faster than the raster scan order used by MPEG-2. For a 16×16 pixel area there are 8 such squares with sides 16, 14, 12, 10, 8, 6, 4 and 2 pixels respectively. Formally, let's assume that $I_n(i, j)$ is the intensity of pixel (i, j) inside a frame n , dx and dy are the motion vector coordinates for a candidate best match of the macroblock to be encoded in the reference frame and (k, l) are the coordinates of the upper left hand corner of the current macroblock. Furthermore, assume that q indicates the offset of the upper left corner of any inner square from the coordinates (k, l) and m is an offset from the upper left hand corner of any square. Then the SAD difference between the reference and the current macroblocks according to the proposed scanning order can be represented by the following equation:

$$\begin{aligned}
 & SAD_{\text{reference_macroblock} - \text{current_macroblock}} \\
 &= \sum_{q=0}^{q=7} \left[\sum_{m=0}^{m=15-2 \times q} |I_t(k+q+m, l+q) \right. \\
 &\quad - I_{t-1}(k+q+m+dx, l+q+dy)| \\
 &\quad + \sum_{m=0}^{m=15-2 \times q} |I_t(k+(15-q), l+q+m) \\
 &\quad - I_{t-1}(k+(15-q)+dx, l+q+m+dy)| \\
 &\quad + \sum_{m=0}^{m=15-2 \times q} |I_t(k+(15-q)-m, l+(15-q)) \\
 &\quad - I_{t-1}(k+(15-q)-m+dx, l+(15-q)+dy)| \\
 &\quad + \sum_{m=0}^{m=15-2 \times q} |I_t(k+q, l+(15-q)-m) \\
 &\quad \left. - I_{t-1}(k+q+dx, l+(15-q)-m+dy)| \right]. \quad (1)
 \end{aligned}$$

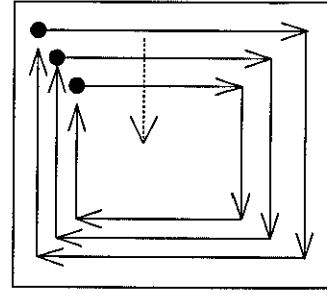


Fig. 2. Spiralling inward scanning order.

The following figure shows the proposed scanning order (Fig. 2):

In the above figure, SAD computation starts from the points depicted as diamonds and potential EJO points are at the corners of the squares of decreasing size for both the reference and current macroblocks. The direction of the SAD computation for the pixels on the sides of squares of decreasing size is shown using arrowheads. The scanning order moves inwards in the macroblocks and the minimum size of the sides of the squares is 2 pixels. The dotted arrowhead shows the direction of the scanning order from the borders of the macroblocks to the centre. In the case that an EJO occurs, motion estimation continues as in the spiral search from the next search point inside the search window of the reference frame. Finally, it has to be noted that the scanning order *has to be the same* for both the current macroblock and its potential best match, since this is a pre-condition for finding exactly the same best matches as the Full Search (FS). If the scan order is different, we risk premature search termination with adverse effects on quality and bit rate.

2.2. Alternating spiralling inward scanning order

The design of this scanning order is based on the idea that the spiralling inward scanning order may be wasting computations due to the fixed direction of the scanning order (top horizontal side of square-right vertical side of square-bottom horizontal side of square-left vertical side of square). In fact, the spiralling inward scanning order will reject faster candidate macroblocks on the basis of horizontal SAD information rather than on the basis of vertical SAD information. This is evident from the order itself, since a candidate macroblock to be rejected in the reference frame on the basis of vertical SAD information will have to wait until the horizontal SAD computations are performed. For this reason, a less biased scheme can be designed which uses horizontal and vertical SAD information for rejection of candidate macroblocks on an alternating basis. Thus in the alternating scanning order case, the scan directions (top horizontal side of

outer square-right vertical side of outer square-bottom horizontal side of outer square-left vertical side of outer square) and (left vertical side of the inner square-bottom horizontal side of the inner square-right vertical side of the inner square-top horizontal side of the inner square) are used. The cycle completes after two iterations and subsequently the outer square becomes inner and the algorithm continues. Again notice that for a 16×16 pixel area there are 8 inner squares with sides 16,14,12,10,8,6,4 and 2 pixels respectively. Formally, let's assume that $I_n(i,j)$ is the intensity of pixel (i,j) inside a frame n , dx and dy are the motion vector coordinates for a candidate best match of the macroblock to be encoded in the reference frame and (k,l) are the coordinates of the upper left hand corner of the current macroblock. Furthermore, assume that q indicates the offset of the upper left hand corner of any inner square from the coordinates (k,l) and m is an offset from the upper left hand corner of any square. Then the SAD difference between the reference and the current macroblocks according to the proposed scanning order can be represented by the following equation:

$SAD_{reference_macroblock-current_macroblock}$

$$\begin{aligned}
 &= \sum_{q=0}^{q=3} \left\{ \left[\sum_{m=0}^{m=15-4 \times q} |I_t(k+2 \times q+m, l+2 \times q) \right. \right. \\
 &\quad - I_{t-1}(k+2 \times q+m+dx, l+2 \times q+dy)| \\
 &\quad + \sum_{m=0}^{m=15-4 \times q} |I_t(k+(15-2 \times q), l+2 \times q+m) \\
 &\quad - I_{t-1}(k+(15-2 \times q)+dx, l+2 \times q+m+dy)| \\
 &\quad + \sum_{m=0}^{m=15-4 \times q} |I_t(k+(15-2 \times q)-m, l+(15-2 \times q)) \\
 &\quad - I_{t-1}(k+(15-2 \times q)-m+dx, l+(15-2 \times q)+dy)| \\
 &\quad + \sum_{m=0}^{m=15-4 \times q} |I_t(k+2 \times q, l+(15-2 \times q)-m) \\
 &\quad \left. - I_{t-1}(k+2 \times q+dx, l+(15-2 \times q)-m+dy)| \right] \\
 &\quad + \left[\sum_{m=0}^{m=15-(4 \times q+2)} |I_t(k+(2 \times q+1), l+(2 \times q+1)+m) \right. \\
 &\quad - I_{t-1}(k+(2 \times q+1)+dx, l+(2 \times q+1)+m+dy)| \\
 &\quad + \sum_{m=0}^{m=15-(4 \times q+2)} |I_t(k+(2 \times q+1)+m, l \\
 &\quad + (15-(2 \times q+1))) - I_{t-1}(k+(2 \times q+1)+m \\
 &\quad + dx, l+(15-(2 \times q+1))+dy)| \\
 &\quad \left. + \sum_{m=0}^{m=15-(4 \times q+2)} |I_t(k+(15-(2 \times q+1)), l \right.
 \end{aligned}$$

$$\begin{aligned}
 &\quad + (15-(2 \times q+1)) - m) - I_{t-1}(k+(15-(2 \times q+1)) \\
 &\quad + dx, l+(15-(2 \times q+1)) - m+dy)| \\
 &\quad + \sum_{m=0}^{m=15-(4 \times q+2)} |I_t(k+(15-(2 \times q+1)) - m, l \\
 &\quad + (2 \times q+1)) - I_{t-1}(k+(15-(2 \times q+1)) \\
 &\quad - m+dx, l+(2 \times q+1)+dy)| \left. \right\}. \quad (2)
 \end{aligned}$$

In the above equation, the first part of the summation computes the SAD for pixels on the sides of squares with side length 16,12,8 and 4 pixels which are visited during the first cycle of the proposed scanning order. Similarly, the second part of the summation computes the SAD for pixels on the sides of squares with side length 14,10,6 and 2 pixels which are visited during the second cycle of the proposed scanning order.

The following figure shows the proposed scanning order (Fig. 3):

As in the previous figure, SAD computation starts from the points depicted as diamonds and potential EJO points are at the corners of the squares of decreasing size inside the reference and current macroblocks. The direction of the SAD computation for the pixels on the sides of squares of decreasing size is shown using arrowheads. The dotted arrowhead shows the direction of the scanning order from the borders of the macroblocks to the centre. Furthermore, the star symbol indicates the position where the cycle is restarted and again the minimum size of the sides of the squares is 2 pixels.

2.3. Horizontal/vertical scanning order

In essence, both the preceding scanning orders attempt to predict the maximum change in the direction of the motion field with zero initial assumptions. This maximum change in the motion field direction will enable faster rejection of candidate macroblocks in the reference frame, thus saving a significant amount of computations. Although both the preceding orders are ideal for online implementations due to the zero initial

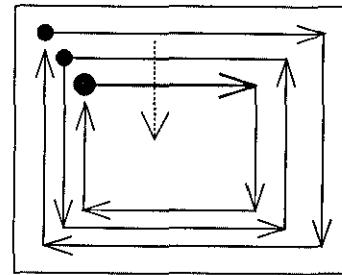


Fig. 3. Alternating spiralling inward scanning order.

assumptions, some computationally economical pre-processing could be beneficial for more accurate estimation of the maximum change in the motion field direction. The trade-off here is that the more accurate the determination of the motion field maximal change, the more pre-processing is required in general. This pre-processing may out-weigh the benefits of computational reduction when the optimal scanning order is found. This lead to the idea of the selection of a horizontal/vertical scanning order for the candidate macroblock in the reference frame by using *only very limited pre-processing*. Independent of our research, a horizontal/vertical scanning order has also been proposed by Kim and Choi [12] but with a much more computationally intense pre-processing phase as will be shown in the section regarding complexity considerations.

Specifically, the first scheme they propose utilises gradient measures on 8×8 block level for finding the best scanning direction per macroblock. The second scheme they propose utilises sorted gradients on a macroblock row level to predict maximal changes in the motion field. The third scheme they propose, further uses sorting of 4×4 sub-block gradients in order to improve the accuracy of the prediction of the maximal change in the motion field. In contrast, our proposed scanning order is determined from examining only the SAD difference between the boundary rows and columns of the macroblock to be encoded and the candidate macroblock in the reference frame. The direction of the maximal SAD difference is then chosen as the scanning direction. A further difference between their scanning orders and the one we propose is the number and the location of the checking points for EJO in the vertical direction. The following figures along with the algorithmic steps show our proposed scanning order:

If $(\max(SAD_{b_cumulative}, SAD_a, SAD_c) = SAD_{b_cumulative})$ use direction *d*
 else if $(\max(SAD_{b_cumulative}, SAD_a, SAD_c) = SAD_c)$ use direction *e*
 else use direction *f*

Step 1: This step is applicable only for the top, bottom, left and right most macroblock rows and columns for the current and reference macroblocks. For notational reasons, let us consider the vertical directions b1–b4 as distinct in this step although they are actually the same. SAD information is computed initially from the top macroblock rows of the macroblock to be encoded and its candidate predictor in the reference frame (direction *a*), the left most and right most columns of the two macroblocks (directions b1–b4) and the bottom rows of the two macroblocks (direction *c*). If the SAD is greater than the minimum SAD found so far along any direction, an EJO can occur at the end of the top most and bottom most rows or in the middle and at the end of the *right most* column. This

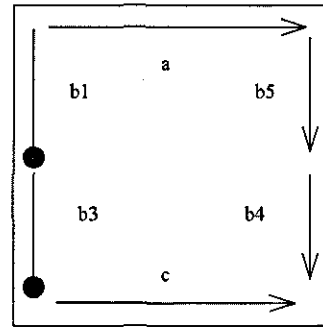


Fig. 4. Step 1 of horizontal/vertical scanning order.

is indicated by the arrowheads in the figure above. The SAD computation is sequential in the above figure since if an EJO does not occur for the direction *a*, the computation for SAD continues in the directions b1–b4 and if we still have not jumped out, we examine SAD across the direction *c*. Notice that there are *only two* potential EJO points in the vertical direction. For the first EJO point, we are essentially adding SADs across the first half of the left-most and right-most columns and then we check for EJO. In the case of the EJO condition not being satisfied, we continue with the SAD computation in the same manner across the second half of the left-most and right-most columns and we repeat the check again. This is indicated by the use star symbols instead of arrowheads in the left most column of the macroblock in Fig. 4. If an EJO has not occurred at the end of the bottom macroblock row, the SADs from directions *a* and *c* and the cumulative SAD from directions b1–b4 are used to determine the scanning order in Step 2.

Step 2: There are three potential scanning orders (directions *d*, *e* and *f*) for the rest of the SAD calculation and they are chosen according to the following scheme:

The following figure shows the potential scanning orders for the current and reference macroblocks according to the outcome of Step 2 (Fig. 5):

In the above figure, the dashed arrowheads show the direction of the scanning order when directions *e* and *f* are chosen. However, when the vertical direction is chosen (direction *d*), we check for EJO in the middle and at the end of the *second* right most macroblock column after we added the corresponding SADs of the *second* left most macroblock column. If an EJO is not found, the algorithm continues with the third left most and right most macroblock columns and so on until we

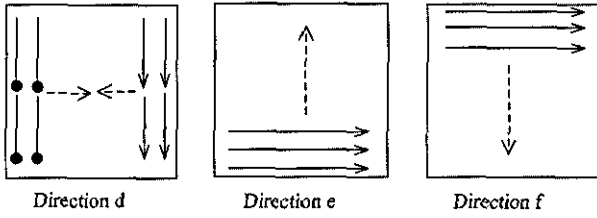


Fig. 5. Potential scanning orders according to the outcome of step 2 of the horizontal/vertical scan method.

either jump out because we found an EJO or we reach the middle of the macroblock. Assume (k, l) are the coordinates of the upper left-most corner of the first inner square in the current macroblock, m and n are the horizontal and vertical offsets from the upper left-most corner of the first inner square and dx and dy are the motion vector coordinates for a candidate best match of the macroblock to be encoded in the reference frame. Step 2 of the proposed scanning order can then be expressed formally as

if direction d is chosen

$$SAD_{reference_macroblock-current_macroblock_{step2}}$$

$$= \left[\sum_{n=1}^{n=7} \sum_{m=1}^{m=7} |I_t(k+n, l+m) - I_{t-1}(k+n+dx, l+m+dy)| \right. \\ \left. + |I_t(k+(15-n), l+m) - I_{t-1}(k+(15-n)+dx, l+m+dy)| \right. \\ \left. + \sum_{n=1}^{n=7} \sum_{m=1}^{m=7} |I_t(k+n, l+8+m) - I_{t-1}(k+n+dx, l+8+m+dy)| \right. \\ \left. + |I_t(k+(15-n), l+8+m) - I_{t-1}(k+(15-n)+dx, l+8+m+dy)| \right]$$

else if direction e is chosen

$$SAD_{reference_macroblock-current_macroblock_{step2}}$$

$$= \left[\sum_{n=14}^{n=1} \sum_{m=1}^{m=14} |I_t(k+m, l+n) - I_{t-1}(k+m+dx, l+n+dy)| \right]$$

else if direction f is chosen

$$SAD_{reference_macroblock-current_macroblock_{step2}}$$

$$= \left[\sum_{n=1}^{n=14} \sum_{m=1}^{m=14} |I_t(k+m, l+n) - I_{t-1}(k+m+dx, l+n+dy)| \right]. \quad (3)$$

Finally, it is worth noting that we can further simplify Step2 with a “rough” knowledge of the dominant motion in a video sequence. For example, Step 2 will enable faster rejection of candidate reference macroblocks for sequences that have predominantly horizontal rather than vertical motion. This is due to the fact that most of the cases will be handled by the first conditional branch and thus two extra comparisons can be saved in most of the cases. Conversely, we can save comparisons for predominantly vertically moving sequences by simply rearranging the order of the conditional statement. We will provide comprehensive results for both predominantly horizontal, vertical and purely mixed sequences in Section 4.

3. Complexity considerations

The issue of complexity has been addressed in the literature mainly from the number of operations point of view and indirectly using metrics related to memory accesses [28]. However, issues like the pre-processing time for specific algorithms have largely been neglected. This pre-processing time can indeed make a motion estimation algorithm slower as compared to MPEG-2 at run time even if there is reduction in the number of operations and the number of memory accesses. For this reason, pre-processing complexity issues are discussed in this section but also actual run times are measured for the three developed algorithms in the following section describing experiments. The closest point of comparison for the developed schemes are the three motion estimation algorithms proposed in [11–12] and their theoretical analysis [30] which may reduce the average number of checking rows per macroblock but they do suffer from pre-processing overhead. For the sake of clarity, the suggested scanning orders are also compared with a fixed scanning order based on dithering of 4×4 sub-blocks as shown in [29]. In this scanning order, there is no pre-processing overhead since the order of checking pixel positions inside the sub-blocks is pre-determined. In the scanning order determination based on 8×8 block gradients in [12], the overhead stems purely from the gradient computation for each block in a macroblock. In the scanning order determination based on sorted macroblock rows [12] or sorted sub-block gradients [11], the overhead stems both from the gradient computation per block but also from the sorting required. Let us examine closely the overhead of gradient computation for the scanning order determination based on 8×8 block gradients in [12]. To compute the gradient of an 8×8 block in the most computationally efficient way, one has to evaluate the pixel gradients at each pixel separately for both the x and y directions and then the sum of the gradient magnitudes along the two directions will be the value of

the gradient for the pixel. This is clearly seen using the following approximations from [11]:

$$|G[f(x, y)]| = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y| \approx |f(x, y) - f(x + 1, y)| + |f(x, y) - f(x, y + 1)|, \quad (4)$$

where $f(x, y)$ is the pixel intensity at position (x, y) , G is the gradient of the pixel intensity and G_x and G_y are the partial pixel gradients across the horizontal and vertical directions respectively.

The above calculation requires 3 additions/subtractions per pixel gradient in the best case and 2 absolute value calculations. For a 256 pixel area, the gradient computation will thus require 768 additions/subtractions and 512 absolute value computations. Subsequently in [12] the block gradient (8×8 pixel area) is computed as the sum of pixel gradients. This will result in further 63 additions/block for a total of 252 additions for the macroblock. Thus the *block* gradient algorithm will cost 1020 additions/subtractions and 512 absolute value computations. The block gradients are subsequently added in pairs and according to the maximum value of these pairs a scanning direction is chosen (right to left, left to right, top to bottom or bottom to top). This further costs 4 additions and 3 comparisons, for a total cost of 1024 additions/subtractions, 512 absolute value computations and 3 comparisons.

The scanning order determination based on sorted macroblock rows in [12], chose the scanning direction in exactly the same manner as the first algorithm on the same paper [12]. The only difference is that after the scanning direction has been chosen, there is the extra overhead of sorting the *macroblock* rows or columns. The fastest sorting algorithms require $O(n \log_n)$ comparisons with n the size of input (in this case 16 macroblock rows or columns), thus requiring 64 comparisons for the sorting phase. Furthermore, 16 more additions are required for forming the gradients of the macroblock rows/columns from the partial gradients on the corresponding blocks, resulting in total pre-processing cost of 1040 additions/subtractions, 512 absolute value computations and 67 comparisons.

In the scanning order determination based on sorted sub-block gradients [11], the scanning direction is chosen in terms of sorted sub-blocks rather than in terms of sorted scan lines in a macroblock. The argument here is that sorted sub-blocks inside a macroblock can better pinpoint the direction of the maximal change in the local motion field as compared to sorted scan lines, thus operations can be saved in the process of finding the best macroblock match in the reference frame. Let us consider the computational overhead of such a scheme: For a 256 pixel area, the gradient computation will require 768 additions/subtractions and 512 absolute value computations as in the algorithms above. To group these gradients into 16 sub-blocks,

where the gradient of each sub-block is the sum of the gradients of its constituent pixels we need 240 additions for a total of 1008 additions/subtractions and 512 absolute value computations for the sub-block gradient formation phase. To sort 16 sub-blocks, we further need 64 comparisons for a total pre-processing cost of 1008 additions/subtractions, 512 absolute value computations and 64 comparisons.

It is clear that although the average number of *macroblock* rows examined by the three aforementioned schemes may indeed be less than the number of rows examined in fixed raster scan orders like the one used in TM5 of MPEG-2 [1], the pre-processing cost of these schemes in terms of operations is higher than the three scanning orders proposed. In particular, the spiralling inward and the alternating spiral schemes have zero pre-processing cost, while the content dependent horizontal/vertical scanning order of the third proposed scheme only requires 120 additions/subtractions for SAD calculation and 4 comparisons, since only the macroblock boundary row and column pixels are used for determining the scanning order. As compared to the dithering scanning order [29], the spiralling inward and the alternating spiral schemes have the same pre-processing overhead (none), while the benefits of the content dependent horizontal/vertical scanning order in terms of predicting the direction of the motion field have to be weighted against the small pre-processing overhead required (124 operations).

4. Experiments and conclusion

The proposed scanning orders were extensively tested for motion estimation performance in a variety of commonly used video sequences exhibiting different motion characteristics. The test sequences "Deadline", "Mother and Daughter (MaD)" and "Students" can be categorised as slow motion sequences. The sequence "Bowling" was also a slow motion sequence but in addition it contained objects that moved forward (zooming) and downward (vertically). "Tennis" and "Paris" were fast paced sequences but the difference among them was that there were objects moving horizontally in "Paris", while in "Tennis" they moved vertically. Finally, "Rotating City" contained a large fast motion area and it involved zooming and panning of the camera. Each sequence consisted of 50 frames except "Rotating City" which only consisted of 35 frames. The performance evaluation in terms of speed-ups was also performed for a variety of search windows with sizes ranging from ± 7 to ± 63 pixels. To facilitate comparisons, the computational savings in terms of average number of rows examined in the motion estimation process are presented, operation count ratio per macroblock and finally the actual run time speed-up

ratio including the pre-processing stage of motion estimation. This stage-by-stage presentation of results is intentional because although the average number of examined macroblock rows is commonly used in the literature [11–12], such a metric cannot account for pre-processing costs. Furthermore even if the pre-processing cost is accounted for in metrics such as the operation count ratio, issues like regularity in memory accesses still remain unaccounted for. The metric that encompasses all the factors affecting the performance of different scanning orders is therefore the actual run time speed-up ratio. It should be noted however that both the average number of macroblock rows examined and the operation count ratio can still be useful performance indicators for multiple pass coding schemes since the motion estimation statistics of the first pass can significantly reduce costs in subsequent passes. The experiments were performed on a Pentium-2 processor at 1 GHz. We also compare our schemes with other well-known work in the literature and we use the following acronyms in the graphs:

- *Spiral* denotes the spiralling inward scanning order we propose.
- *Alt-Spiral* denotes the alternating spiral scanning order we propose.
- *Vbt* and *Tbv* denote the vertical/horizontal and the horizontal/vertical scanning orders we propose. In fact Vbt stands for the vertical-bottom-top scanning order and Tbv for the top-bottom-vertical order.
- *Kim1* denotes the second scanning algorithm in Ref. [12].
- *Kim2* denotes the scanning algorithm based on complexity using 4×4 sub-blocks in Ref. [11].
- *Kim3* denotes the first scanning algorithm in Ref. [12].
- *Dithering* denotes a fixed scanning order based on 4×4 sub-blocks as in Ref. [29].
- *MPEG-2* denotes the fixed scanning order (left to right-top to bottom) that is used in the standard (TM5).

In Table 1A, the average number of examined rows per macroblock with respect to the window size is shown. This average is over all the candidate positions inside a search window in the reference frame. It can be seen that the average number of rows examined decreases as the search window increases. This is expected since the bigger the search window, the more candidate positions for the best match in the reference frame will be rejected after only a small number of macroblock rows is examined, thus bringing the average down. Obviously, the gains of the proposed methods depend both on the sequence motion characteristics as well as the window size. On the average, gains of 7.14% are observed across all window sizes in Table 1B for the proposed scanning orders as compared to the MPEG-2 raster scan. Furthermore, 9.87% gains over MPEG-2

for other popular adaptive scanning orders requiring pre-processing are also observed in the same table.

For comparison purposes, results for the dithering scanning order are also presented, which is a fixed scanning order with average gains 2.41% over the MPEG-2 scanning order in Table 1B. It has to be noted that the proposed schemes consistently outperform dithering and they are highly competitive with respect to other adaptive schemes requiring much more pre-processing as shown in the tables. In fact, it can be seen that the average gain of other adaptive schemes requiring pre-processing over the suggested schemes is less than 3% and the biggest differences occur at small window sizes.

The effects of pre-processing in terms of speed-up ratios but also in terms of actual run times complexity (in seconds) are shown in Table 2B and A, respectively. The actual run times refer to the total time needed for encoding 50 (35 frames for the rotating city sequence) of each of the tested sequences and includes the pre-processing time needed for the motion estimation. The speed-up ratios denote the percentage gain with respect to the actual run time. To facilitate comparisons, the low complexity scanning order part of these tables refers to schemes that require no or minimal pre-processing, while the high complexity scanning order part refers to other schemes in the literature that require significant pre-processing. It is evident from Table 2A and B that across different size windows, the proposed scanning orders have a higher speed-up ratio with respect to the MPEG-2 raster scan order by 0.12 on the average, while other adaptive schemes in the literature have a lower speed-up ratio by 39.7 times. Even the fixed dithering scan order has a lower speed-up ratio with respect to MPEG-2 by 0.78 on the average due to the irregularity in memory access patterns [13–14]. From the results, it can be seen that the proposed scanning orders consistently outperform both the dithering and the adaptive schemes in the literature and bigger differences in the performance benefits occur for larger window sizes. This clearly reveals a trade-off between fine complexity localisation from the adaptive scanning orders in the literature and the pre-processing overhead imposed from such methods. As the window size increases, complexity localisation becomes less important in motion estimation schemes involving Early Jump Outs, while the pre-processing overheads remain. Thus, for scanning orders with pre-processing cost far outweighing the complexity localisation benefits, the run time performance can actually degrade and this effect is amplified with larger window sizes. In this light, run time improvements of lower complexity schemes like the ones proposed make them very attractive for real time applications. Table 3 shows the total number of operations per macroblock for the scanning orders examined and includes the effects of pre-processing as

Table 1

Sequence	Window size (\pm)	Low complexity scanning orders						High complexity scanning orders		
		MPEG2	Spiral	ALT-Spiral	VBT	TBV	Dithering	Kim1	Kim2	Kim3
(A) Average number of examined rows per macroblock										
Bowling	7	5.23	4.46	4.54	4.57	4.57	4.85	4.36	4.13	4.55
	15	3.94	3.39	3.43	3.45	3.45	3.69	3.35	3.23	3.54
	23	3.26	2.82	2.84	2.87	2.87	3.07	2.80	2.72	2.97
	31	2.82	2.45	2.47	2.47	2.47	2.67	2.44	2.39	2.59
	63	1.87	1.65	1.65	1.66	1.66	1.78	1.65	1.63	1.74
Deadline	7	3.08	2.77	2.79	2.78	2.78	3.01	2.60	2.39	2.81
	15	2.33	2.10	2.11	2.10	2.10	2.28	2.04	1.93	2.19
	23	1.97	1.80	1.80	1.80	1.80	1.93	1.77	1.71	1.89
	31	1.77	1.61	1.61	1.61	1.61	1.74	1.62	1.57	1.72
	63	1.32	1.21	1.21	1.20	1.20	1.32	1.25	1.23	1.31
MaD	7	5.84	5.19	5.29	5.32	5.32	5.61	5.10	4.80	5.31
	15	4.57	4.03	4.10	4.11	4.11	4.41	4.06	3.88	4.23
	23	3.82	3.37	3.42	3.44	3.44	3.70	3.43	3.31	3.58
	31	3.36	2.95	2.98	3.00	3.00	3.24	3.00	2.92	3.14
	63	2.23	1.97	1.98	1.99	1.99	2.16	2.02	1.99	2.11
Paris	7	3.06	2.73	2.75	2.73	2.73	2.94	2.52	2.31	2.69
	15	2.34	2.08	2.10	2.09	2.09	2.26	2.00	1.90	2.11
	23	2.03	1.81	1.82	1.81	1.81	1.96	1.78	1.71	1.87
	31	1.85	1.66	1.67	1.66	1.66	1.79	1.66	1.61	1.72
	63	1.47	1.34	1.35	1.35	1.35	1.44	1.37	1.33	1.40
Students	7	3.25	2.89	2.92	2.91	2.91	3.10	2.71	2.46	2.90
	15	2.45	2.17	2.18	2.17	2.17	2.34	2.11	1.98	2.25
	23	2.05	1.83	1.82	1.82	1.82	1.98	1.80	1.71	1.91
	31	1.82	1.64	1.64	1.63	1.63	1.76	1.62	1.56	1.71
	63	1.34	1.23	1.23	1.23	1.23	1.32	1.25	1.23	1.31
Tennis	7	6.26	5.89	5.98	5.98	5.98	6.15	5.64	5.52	5.96
	15	5.29	4.95	4.98	5.00	5.00	5.15	4.71	4.64	5.08
	23	4.78	4.45	4.48	4.49	4.49	4.64	4.24	4.18	4.60
	31	4.49	4.17	4.19	4.20	4.20	4.33	3.98	3.92	4.33
	63	3.69	3.41	3.42	3.43	3.43	3.55	3.27	3.24	3.56
Rotating city	7	10.73	10.14	10.46	10.52	10.52	10.77	10.64	10.57	10.60
	15	8.88	8.38	8.59	8.60	8.60	8.87	8.75	8.70	8.76
	23	7.56	7.11	7.26	7.24	7.24	7.52	7.41	7.37	7.43
	31	6.66	6.25	6.35	6.32	6.32	6.61	6.50	6.47	6.53
	63	4.33	4.29	4.33	4.30	4.30	4.53	4.46	4.13	4.53
(B) Percentage gain in number of examined rows										
Bowling	7	0%	15%	13%	13%	13%	7%	17%	21%	13%
	15	0%	14%	13%	13%	13%	6%	15%	18%	10%
	23	0%	14%	13%	12%	12%	6%	14%	17%	9%
	31	0%	13%	12%	13%	13%	6%	13%	15%	8%
	63	0%	12%	12%	11%	11%	4%	12%	13%	7%
Deadline	7	0%	10%	9%	10%	10%	2%	15%	22%	9%
	15	0%	10%	9%	10%	10%	2%	12%	17%	6%
	23	0%	9%	9%	9%	9%	2%	10%	13%	4%
	31	0%	9%	9%	9%	9%	1%	8%	11%	3%
	63	0%	8%	8%	9%	9%	0%	5%	7%	1%
MaD	7	0%	11%	9%	9%	9%	4%	13%	18%	9%
	15	0%	12%	10%	10%	10%	3%	11%	15%	7%
	23	0%	12%	11%	10%	10%	3%	10%	13%	6%
	31	0%	12%	11%	11%	11%	4%	11%	13%	7%
	63	0%	11%	11%	11%	11%	3%	10%	11%	5%
Paris	7	0%	11%	10%	11%	11%	4%	18%	24%	12%
	15	0%	11%	10%	11%	11%	3%	14%	19%	10%
	23	0%	11%	10%	11%	11%	4%	12%	15%	8%
	31	0%	10%	10%	10%	10%	4%	11%	13%	7%
	63	0%	9%	8%	9%	9%	2%	7%	10%	5%
Students	7	0%	11%	10%	11%	11%	5%	17%	24%	11%
	15	0%	11%	11%	11%	11%	4%	14%	19%	8%
	23	0%	11%	11%	11%	11%	4%	12%	17%	7%

Table 1 (continued)

Sequence	Window size (\pm)	Low complexity scanning orders						High complexity scanning orders		
		MPEG2	Spiral	ALT-Spiral	VBT	TBV	Dithering	Kim1	Kim2	Kim3
Tennis	31	0%	10%	10%	10%	10%	3%	11%	14%	6%
	63	0%	9%	9%	9%	9%	1%	7%	9%	3%
	7	0%	6%	5%	4%	4%	2%	10%	12%	5%
	15	0%	7%	6%	6%	6%	3%	11%	12%	4%
	23	0%	7%	6%	6%	6%	3%	11%	12%	4%
Rotating city	31	0%	7%	7%	6%	6%	4%	11%	13%	4%
	63	0%	8%	7%	7%	7%	4%	11%	12%	3%
	7	0%	6%	3%	2%	2%	0%	1%	1%	1%
	15	0%	6%	3%	3%	3%	0%	1%	2%	1%
	23	0%	6%	4%	4%	4%	1%	2%	2%	2%
	31	0%	6%	5%	5%	5%	1%	2%	3%	2%
	63	0%	1%	0%	1%	1%	-4%	-3%	5%	-4%
Average (%)		0	7.76	6.92	6.94	6.94	2.41	10.35	13.40	5.85

Table 2

sequence	Window size (\pm)	Low complexity scanning orders						High complexity scanning orders		
		MPEG2	Spiral	ALT-spiral	VBT	TBV	Dithering	Kim1	Kim2	Kim3
(A) Total encoding time (s)										
Bowling	7	6.10	5.75	5.72	5.44	5.46	7.08	167	158	127
	15	17.26	15.70	15.83	14.68	14.77	20.23	702	668	534
	23	31.79	28.79	28.92	26.73	26.67	38.41	1558	1479	1181
	31	48.68	44.23	44.41	40.81	40.88	61.95	2736	2599	2075
	63	131.17	119.83	119.76	115.86	109.38	191.35	10150	9402	7503
Deadline	7	4.44	4.32	4.29	4.09	4.05	7.08	170	162	131
	15	11.45	10.86	10.93	10.18	10.19	32.03	718	684	545
	23	21.04	19.85	19.97	18.50	18.32	60.68	1591	1517	1211
	31	32.92	31.31	31.44	29.00	28.84	61.95	2806	2666	2127
	63	97.83	93.53	93.53	86.94	85.98	191.35	10150	9683	7717
MaD	7	7.06	6.91	6.98	6.62	6.63	11.73	173	164	133
	15	20.51	19.55	19.71	18.40	18.40	60.47	721	692.38	550.08
	23	38.00	35.72	36.11	33.19	33.26	67.41	1595	1529	1215
	31	58.59	55.01	55.38	50.76	50.67	105.02	2801	2683	2133
	63	155.12	144.10	144.79	131.40	130.64	283.04	10123	9696	7702
Paris	7	4.40	4.27	4.25	4.06	3.96	6.94	169	161	131
	15	11.35	10.62	10.83	9.87	9.86	20.13	717	680	544
	23	21.27	19.78	19.96	18.26	18.24	38.80	1592	1509	1224
	31	34.02	31.71	32.00	29.32	29.04	63.15	2803	2657	2154
	63	106.93	100.90	101.13	92.45	92.01	205.54	10179	9660	7715
Students	7	4.60	4.42	4.47	4.19	4.15	7.26	171	162	132
	15	11.80	11.14	11.21	10.28	10.30	20.57	722	685	548
	23	21.45	20.19	20.27	18.49	18.40	38.90	1603	1520	1229
	31	33.36	31.41	31.50	28.73	28.54	61.70	2820	2672	2160
	63	98.90	93.66	94.01	85.91	84.97	189.68	10182	9698	7736
Tennis	7	6.67	6.53	6.61	6.31	6.31	11.21	148	140	114
	15	21.37	20.35	20.60	19.26	19.42	37.87	626	591	478
	23	43.13	40.69	41.33	38.27	38.42	77.74	1382	1305	1052
	31	71.89	67.62	68.43	63.32	63.60	130.19	2430	2287	1846
	63	235.93	220.57	223.62	206.99	205.51	429.98	8672	8199	6595
Rotating city	7	13.72	13.66	13.77	13.03	12.87	22.40	236	224	179
	15	45.36	44.20	44.53	41.36	41.18	76.32	984	942	751
	23	87.74	84.57	85.46	77.60	77.33	149.17	2195	2098	1649
	31	138.60	132.36	133.03	120.45	120.01	236.22	3869	3698	2903
	63	392.46	365.45	368.39	323.32	322.97	673.13	14391	13799	10952

Table 2 (continued)

sequence	Window size (\pm)	Low complexity scanning orders						High complexity scanning orders		
		MPEG2	Spiral	ALT-spiral	VBT	TBV	Dithering	Kim1	Kim2	Kim3
(B) Speedup ratio										
Bowing	7	1.00	0.94	0.94	0.89	0.90	1.16	27.39	26.06	20.90
	15	1.00	0.91	0.92	0.85	0.86	1.17	40.70	38.72	30.99
	23	1.00	0.91	0.91	0.84	0.84	1.21	49.01	46.54	37.18
	31	1.00	0.91	0.91	0.84	0.84	1.27	56.21	53.40	42.64
	63	1.00	0.91	0.91	0.88	0.83	1.46	77.39	71.68	57.21
Deadline	7	1.00	0.97	0.97	0.92	0.91	1.59	38.36	36.59	29.65
	15	1.00	0.95	0.95	0.89	0.89	2.80	62.71	59.80	47.64
	23	1.00	0.94	0.95	0.88	0.87	2.88	75.62	72.11	57.56
	31	1.00	0.95	0.96	0.88	0.88	1.88	85.24	81.01	64.63
	63	1.00	0.96	0.96	0.89	0.88	1.96	103.76	98.99	78.89
MaD	7	1.00	0.98	0.99	0.94	0.94	1.66	24.54	23.35	18.86
	15	1.00	0.95	0.96	0.90	0.90	2.95	35.16	33.76	26.82
	23	1.00	0.94	0.95	0.87	0.88	1.77	41.98	40.25	31.98
	31	1.00	0.94	0.95	0.87	0.86	1.79	47.82	45.80	36.41
	63	1.00	0.93	0.93	0.85	0.84	1.82	65.26	62.51	49.65
Paris	7	1.00	0.97	0.97	0.92	0.90	1.58	38.55	36.68	29.86
	15	1.00	0.94	0.95	0.87	0.87	1.77	63.24	59.92	48.02
	23	1.00	0.93	0.94	0.86	0.86	1.82	74.88	70.98	57.59
	31	1.00	0.93	0.94	0.86	0.85	1.86	82.41	78.11	63.34
	63	1.00	0.94	0.95	0.86	0.86	1.92	95.19	90.34	72.15
Students	7	1.00	0.96	0.97	0.91	0.90	1.58	37.32	35.43	28.72
	15	1.00	0.94	0.95	0.87	0.87	1.74	61.26	58.09	46.46
	23	1.00	0.94	0.94	0.86	0.86	1.81	74.74	70.87	57.33
	31	1.00	0.94	0.94	0.86	0.86	1.85	84.54	80.10	64.75
	63	1.00	0.95	0.95	0.87	0.86	1.92	102.96	98.06	78.23
Tennis	7	1.00	0.98	0.99	0.95	0.95	1.68	22.33	21.10	17.10
	15	1.00	0.95	0.96	0.90	0.91	1.77	29.31	27.70	22.39
	23	1.00	0.94	0.96	0.89	0.89	1.80	32.05	30.27	24.41
	31	1.00	0.94	0.95	0.88	0.88	1.81	33.81	31.82	25.68
	63	1.00	0.93	0.95	0.88	0.87	1.82	36.76	34.76	27.96
Rotating city	7	1.00	1.00	1.00	0.95	0.94	1.63	17.22	16.37	13.08
	15	1.00	0.97	0.98	0.91	0.91	1.68	21.70	20.79	16.57
	23	1.00	0.96	0.97	0.88	0.88	1.70	25.02	23.92	18.80
	31	1.00	0.95	0.96	0.87	0.87	1.70	27.92	26.69	20.95
	63	1.00	0.93	0.94	0.82	0.82	1.72	36.67	35.16	27.91
Average		1								
Speedup			+0.08	+0.08	+0.15	+0.15	-0.78	-44.73	-42.12	-32.25

Table 3
Cost of operations for motion estimation per macroblock for various scanning orders

Scanning orders	Cost of operations	Operation count ratios with respect to MPEG-2
MPEG-2 raster scan	512	1
Dithering	512	1
Kim1	2051	4 times
Kim2	2131	4.16 times
Kim3	2096	4.09 times
Spiral	512	1
Alt-spiral	512	1
Vbt and Tbv	636	1.24 times

well as the actual motion estimation operations. In this table, the raster scan order used in MPEG-2 requires 256 absolute value computations and 240 additions for

SAD estimation. Further 16 comparisons are required in the EJO points for a total of 512 operations for the MPEG-2 scanning order which does not require pre-processing overhead. The operation count ratios in this table are with respect to the MPEG-2 scan order and all operations are assigned equal weighting. It can be seen from the results that the operation count ratio increases only by 0.24 in the proposed adaptive horizontal/vertical scanning order, it does not increase for the fixed order schemes proposed (spiralling inwards and alternating spiralling inwards) and it increases by 4 or more times in other adaptive schemes in the literature.

Finally, Table 4 shows the average PSNR values per window size with a given channel and frame rate for each tested sequence. It has to be noted that all the tested scanning orders give exactly the same PSNR values (per frame and on the average) for a given window size, channel and frame rates since they find the

Table 4
Average PSNR in various window sizes for all tested scanning orders

Sequence	Channel bit rate(bits/s)	Frame rate(frame/s)	Window size	MSE	PSNR (dB)
Bowing	4,000,000	25	7	1.82	48.96
			15	1.82	48.96
			23	1.80	49.07
			31	1.82	48.96
Deadline	4,000,000	25	63	1.83	48.94
			7	3.57	44.58
			15	3.58	44.58
			23	3.57	44.58
MaD	5,000,000	25	31	3.58	44.57
			63	3.58	44.56
			7	2.69	46.20
			15	2.69	46.20
Paris	5,000,000	25	23	2.70	46.19
			31	2.71	46.19
			63	2.71	46.17
			7	3.99	43.94
Students	5,000,000	25	15	4.00	43.93
			23	3.99	43.94
			31	3.99	43.94
			63	4.00	43.93
Tennis	5,000,000	25	7	2.28	47.49
			15	2.28	47.49
			23	2.28	47.48
			31	2.28	47.49
Rotating city	5,000,000	25	63	2.28	47.48
			7	6.22	41.41
			15	6.17	41.47
			23	6.17	41.47
			31	6.17	41.47
			63	6.18	41.46
			7	10.32	38.28
			15	10.32	38.49
			23	10.10	38.59
			31	9.70	38.78
			63	7.22	39.55

same best matches in the reference frame. From the results, it can also be observed that the PSNR does not necessarily increase with bigger window size. This is a side effect of choosing the best predictor for the macroblock to be encoded based on the minimum SAD metric only and irrespective of how similar the predictor is to the macroblock to be encoded. This entrapment to local minima has ramifications in both the MSE and PSNR metrics as shown in Table 4.

In conclusion, the three proposed scanning orders (spiralling inwards-alternating spirals and horizontal/vertical) achieve average gains of 7.14% in terms of examined macroblocks rows as compared to the MPEG-2 raster scan order and are within 3% in terms of average gains of other schemes in the literature requiring much heavier pre-processing. They have the same or very similar operation count ratios as compared to MPEG-2, in contrast to other schemes involving pre-processing which increase the operation count ratios more than 4 times. In terms of run time performance,

the proposed scanning orders increase the speed-up ratio by 0.12 on the average with respect to MPEG-2, as compared to pre-processing schemes that decrease it by 39.7 times on the average. Finally, the proposed schemes consistently outperform the dithering scanning order in the average number of macroblock rows and run time performance metrics, while they have the same or very similar performance in terms of operation counts. For these reasons, the proposed scanning orders could be very attractive for reduced complexity initial estimation of the motion field direction for one pass coding schemes in real time applications.

5. Summary

Scanning orders have been greatly overlooked in the video coding literature in the context of motion estimation, although they have been very successfully employed for increasing the compression efficiency in the coding of DCT coefficients. The design of efficient scanning orders, which reduce the computational cost of the motion estimation, essentially entails complexity localisation in the motion field direction of the reference frame. This complexity localisation is unfortunately a computationally intensive process which may render such scanning orders unsuitable for real time video coding system implementations. In this paper, three low complexity scanning orders of similar performance are proposed that are very competitive in terms of the operation count ratio metric with respect to the MPEG-2 raster scan order, show improvements of 7.14% on the average with respect to the number of examined macroblock rows metric and they also show an increase in speed-up ratio of 0.12 on the average as compared to the standard. As compared to other work in the literature, the proposed scanning orders require one fourth of the operation count ratio and show an increase in the speed-up ratio of 45 times on the average.

References

- [1] MPEG software: <http://www.mpeg.org/MPEG/MSSG/>
- [2] Li R, Zeng B, Liou ML. A new 3 step search Algorithm for Block Motion Estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 1994;4(4):438–43.
- [3] Jain JR, Jain AK. Displacement measurement and its application in interframe image coding. *IEEE Transactions on Communications* 1981;COM 29:1799–806.
- [4] Pennebaker W, Mitchell J. *JPEG still image data compression standard*. New York: Van Nostrand Reinhold; 1994.
- [5] Ghanbari M. The cross-search algorithm for motion estimation. *IEEE Transactions on Communications* 1990;38:950–3.
- [6] Lai-man Po, Wing-Chung Ma. A novel four step-search algorithm for fast block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 1996;6:313–7.

- [7] Liu LK, Feig E. A block based gradient descent search algorithm for block motion estimation in video coding. *IEEE Transactions on Circuits and Systems for Video Technology* 1996;6:419–22.
- [8] Tham JY, Ranganath S, Ranganath M, Kassim AA. A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 1998;8:369–77.
- [9] Tourapis AM, Au OC, Liou M. Highly efficient predictive zonal algorithms for fast block matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 2002;12:934–47.
- [10] Chun-Ho Cheung, Lai-Man Po. Generalised partial distortion search algorithm for block matching motion estimation. Thessaloniki, Greece: IEEE ICIP; 2001.
- [11] Kim JN, Byun SC, Kim YH, Ahn BH. Fast full search motion estimation using early detection of impossible candidate vectors. *IEEE Transactions on Signal Processing* 2002;50(9):2355–65.
- [12] Jong-Nam Kim, Tae-Sun Choi. A fast full search motion estimation algorithm using representative pixels and adaptive matching scan. *IEEE Transactions on Circuits and Systems for Video Technology* 2000;10:1040–8.
- [13] Lengwehasatit K, Ortega A. Probabilistic partial distortion fast matching for motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 2001;11(2):139–52.
- [14] Langwehasatit K. Complexity Distortion trade-offs in image and video compression. PhD Thesis, University of Southern California, May 2000.
- [15] Eckart S, Fogg C. ISO/IEC MPEG-2 software video codec. *Proceedings of SPIE* 1995;2419:100–18.
- [16] ITU-T Recommendation H263 Software Implementation. Digital Video Coding Group, Telenor R&D, 1995.
- [17] Kim JN, Choi TS. Computational Reduction using UESA adaptive partial sum form gradient magnitude for fast motion estimation. *Proceedings of PCS* 1999. p. 107–11.
- [18] Li W, Salari E. Successive elimination algorithm for motion estimation. *IEEE Transactions on Image Processing* 1995;4:105–7.
- [19] DeOliveira GC, Alcaim A. On fast motion compensation algorithms for video coding, in *Proceedings of PCS* 1997. p. 467–72.
- [20] Lu JY, Wu KS, Lin JC. Fast full search in motion estimation by hierarchical use of Minkowski's inequality (HUMI). *Pattern Recognition* 1998;31:945–52.
- [21] Coban MZ, Mersereau RM. A fast exhaustive search algorithm for rate constrained motion estimation. *IEEE Transactions on Image Processing* 1998;9:769–73.
- [22] Wang HS, Mersereau RM. Fast algorithms for the estimation of motion vectors. *IEEE Transactions on Image Processing* 1999;8:435–8.
- [23] Gao XQ, Duanmu CJ, Zou CR. A multilevel successive elimination algorithm for block matching motion estimation. *IEEE Transactions on Image Processing* 2000;9:501–4.
- [24] Oh TM, Kim YR, Hong WG, Ko SJ. A fast full search motion estimation algorithm using the sum of partial norms. *Proceedings of ICCE* 2000. p. 236–7.
- [25] Naito Y, Miyazaki T, Kuroda I. A fast full search motion estimation method for programmable processors with a multiply accumulator. *Proceedings of ICASSP* 1996. p. 3221–4.
- [26] Lin YC, Tai SC. Fast full search block matching for motion compensated video compression. *IEEE Transactions on Communications* 1997;45:527–31.
- [27] Do VL, Yun KY. A low power VLSI architecture for full search block matching motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 1998;8:393–8.
- [28] Patterson DA, Hennesy JL. *Computer architecture: a quantitative approach*. 2nd ed. Los Altos, CA: Morgan-Kaufman Publishers; 1996.
- [29] Cheung CK, LaiMan PO. Normalised partial distortion search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology* 2000;10(3):417–22.
- [30] Tao B, Orchard MT. Gradient Based residual variance modelling and its Applications to Motion Compensated video coding. *IEEE Transactions on Image Processing* 2001;10(1):24–35.

Group PC: Published National and International Conference papers on Parallelism and the Software-Hardware Interface in Embedded Systems

PC1:

V. A. Chouliaras, J. L. Nunez, 'A Scalar Coprocessor for accelerating the G723.1 and G729A Speech Coders', proceedings of the IEEE International Conference on Consumer Electronics (ICCE 2003), Los Angeles, California, USA, ISBN:0-7803-8838-0

This paper presents the results of a first approach in developing a systematic methodology for designing custom ISA extensions for telecom applications. The target was the International Telecommunications Union (ITU-T) speech coding standards G.723.1 and G.729.A. The study profiled the workloads and identified a number of fixed-point extensions that demonstrated a significant reduction in the dynamic instruction count of both speech coders. Subsequently, a scalar coprocessor microarchitecture was proposed to encapsulate these new ISA extension, in the context of a high-performance, dual-issue RISC microprocessor

PC2:

V. A. Chouliaras, J. L. Nunez-Yanez, S. Agha, 'Silicon Implementation of a Parametric Vector Datapath for real-time MPEG2 encoding', Proceedings of the IASTED (SIP) 2004, Honolulu, Hawaii, USA, ISBN: 0-88986-442-X

This contribution extends the methodologies developed over the past few years at the Electronic Systems Design Group through targeting the MPEG-2 TM5 video coding standard. A characteristic of the TM5 reference implementation is the abundance of data and thread-level parallelism which is unfortunately, left to the system architect to discover and recover. In particular, the block-matching inner-function of the ME algorithm, function *dist1*, is described in a mostly sequential way thus making automatic vectorization virtually impossible. The paper follows the systematic methodologies established by the author to characterize the workload, identify the compute-intensive areas of the TM5 implementation, parallelize (through vectorization) and subsequently, implement custom vector instruction ISA extensions. These extensions were encapsulated in a parametric vector accelerator attached to an open-source, configurable, extensible RISC CPU. Finally, three VLSI implementations, for a number of vector register file configurations, were undertaken on a high-performance 0.13 μm , 8-copper layer CMOS process.

PC3:

V. A. Chouliaras, J. L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso '*A multi-standard video coding accelerator based on a vector architecture*', Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2005), Las Vegas, Nevada, USA, ISBN: 07803-8839-9

The work of the previous contribution is expanded to accommodate the very latest video coding standards such as MPEG-4 (integer-based XViD implementation) and H264 (ST Microelectronics proprietary implementation, developed for very low power, portable, consumer products). The same parallelization methodology as in PC2 was applied and significant reductions in the dynamic instruction count of the applications were achieved. The contribution concludes with a suitably detailed microarchitecture of a parametric vector accelerator tightly coupled to a configurable, extensible 32-bit RISC CPU.

PC4:

V. A. Chouliaras, J. A. Flint, Y. Li, '*Parametric Data-Parallel architectures for TLM acceleration*', Proceedings of the 3rd International Conference on Computational Electromagnetics and Its Applications (ICCEA), Nov. 1-4 2004, Beijing, China

This work quantified the benefits of exploiting the DLP in an electromagnetic modelling 3D TLM kernel. Using the infrastructure developed for consumer embedded applications, the joint study with Dr. James Flint started by re-writing the SCN-TLM code in a vectorized form first and then, threading (statically assigning to distinct CPU contexts) the data-parallel sections. A number of experiments were then conducted which quantified with precision the algorithmic benefit (dynamic instruction count reduction) of this application when configured to run on a parametric, vector-floating-point processor.

PC5:

V. A. Chouliaras, J. L. Nunez-Yanez, T. R. Jacobs and Ashwin K. Kumaraswamy, '*Configurable Multiprocessors for high-performance MPEG-4 video coding*', Proceedings of the IEEE Annual Symposium on VLSI, May 11-12 2005, Tampa, Florida, USA.

A number of research groups and major industrial vendors have identified and exploited DLP to a certain extent through ISA extensions, streaming memory systems or a combination thereof. However, very few researchers have touched on TLP in video workloads and this is one of the areas where the Electronic Systems Design Group at Loughborough University has been particularly successful. This contribution

targets the MPEG-4 (XViD) video coding standard implementation and details the development of a custom PRAM simulator, the threading process of the MPEG-4 implementation and evaluates the theoretical performance of the parallelized XViD encoder on the PRAM machine. The results clearly demonstrate a significant reduction in the dynamic instruction count for each CPU context. The paper presents a VLSI macrocell consisting of two modified Leon-2 embedded CPUs each of which includes a custom, hardware-based barrier mechanism and has coherent level 1 data caches. The multiprocessor interconnect is a single 32-bit wide AMBA bus.

PC6:

Ashwin K. Kumaraswamy, V. A. Chouliaras, T. R. Jacobs, and J. L. Nunez-Yanez, '*System-on-Chip Design Framework (SDF) unifying Specification Capture and Design Modelling*', Proceedings of the 2005 Electronic Design Processes (EDP) Workshop, April 6-8, Monterey Beach Hotel, Monterey, California, USA.

The development of the parallelized algorithms and the vector and multi-threaded microarchitectures required to execute them efficiently has been based on a number of industrial methodologies that are consistently applied in embedded workloads. One major drawback of these methodologies is the lack of automation in the process. Clearly, parallelizing applications at the data and thread level is a time-consuming process that can be executed by an expert human programmer, should time is not an issue, as there are no mainstream vectorizing and parallelizing compilers available to embedded CPU designers. The final form of parallelism, instruction level parallelism, is exposed in the vast majority of applications via the compiler illustrating a degree of maturity in the automation of the compilation process. The same cannot be claimed for the hardware design flow which segments, in an ad-hoc manner, the design space into programmable (CPU-based) and non-programmable (hardwired) domains. The CPU-based domain has been studied quite thoroughly since it devolves the hardware complexity to the compiler whereas the hardwired solutions space is the primary candidate for automatic exploitation via high-level specifications such as SystemC. This contribution is the first report of a major effort in fusing together two technologies, nearly cycle-accurate system-on-chip simulation and UML-based integrated system specification capture. The successful fusing of these processes is a significant milestone for the Electronic Systems Design Group as it will permit the precise and unambiguous specification and automatic implementation of large-scale microarchitecture blocks and sub-systems.

PC7:

V M Dwyer, S Agha and V Chouliaras, '*Low Power Full-Search Block Matching using reduced bit SAD values for early termination*', Proceedings of Mirage 2005 International conference on Computer Vision/Computer Graphics collaboration techniques

This work presents an algorithmic optimization for reducing the power consumption of an MPEG-2 TM5 encoder via using a truncated sum-of-absolute-differences (SAD) metric. Typically, full-search motion estimation relies on the computation of an error term which describes how well a macroblock in the *current* frame matches a macroblock within a search window in the *reference* frame. So far, designers have been utilizing full accuracy for performing the arithmetic of the well known SAD computations. In this work, we propose a modified arithmetic scheme in which only the upper four (most-significant) bits of the luma component of the reference and current blocks participate in the computation of the SAD error term. A correction mechanism is implemented that reverts to full-width arithmetic if certain encoding conditions are met.

PC8:

Tom R. Jacobs, Vassilios A. Chouliaras and Jose L. Nunez, '*A thread and data-Parallel MPEG-4 Video Encoder for a System-On-Chip Multiprocessor*', accepted for oral presentation at the IEEE 16th International Conference on application-specific architectures and processors (ASAP 2005), Samos, Greece, July 23-25 2005

Following the successful exploitation of parallelism at the thread level, this contribution elaborates on a combined thread and data parallel implementation of the MPEG-4 XViD video coding standard. It is shown that both types of parallelism should be exploited for optimal execution on power-conscious consumer appliances and presents a modified VLSI implementation of the multi-processor Leon-2 based platform in which each CPU core incorporates the tightly-coupled vector accelerator implementing a subset of the MPEG-4 (PJ3) and the full set of MPEG-2 (PC2) vector extensions.

PC9:

S. R. Parr, K. Koutsomyti, V. A. Chouliaras, J.L. Nunez, D. J. Mulvaney, '*Configurable Scalar and Vector Coprocessors for accelerating the G.723.1 and G.729.A speech coders*', accepted for oral presentation at the IASTED International Conference on Signal and Image Processing (ACIT-SIP), Novosibirsk, Russia, June 20-24, 2005

This contribution provides substantially more research data to those already published in the IEE Electronics Letter contribution (paper PJ2). In particular, it consolidates all the results for both G.723.1 and G.729.A speech coders encoding and decoding, all test vectors and all optimization methods including combined scalar and vector accelerators. The paper finalizes the programmers' model and proposes a highly detailed parametric vector accelerator with each scalar lane being a 2-way SIMD configuration as the implementation of choice.

Paper PC1: **V. A. Chouliaras, J. L. Nunez**, '*A Scalar Coprocessor for accelerating the G723.1 and G729A Speech Coders*', Proceedings of the IEEE International Conference on Consumer Electronics (ICCE 2003), Los Angeles, California, USA, ISBN:0-7803-8838-0

A Scalar Coprocessor for Accelerating the G723.1 and G729A speech coders

Vassilios A. Chouliaras, Jose. L. Nunez
Department of Electronic and Electrical Engineering
University of Loughborough
Loughborough, Leicestershire LE11 3TU

Abstract: A scalar accelerator that reduces significantly the complexity of the ITU-T G723.1 and G729A speech coders is described. Preliminary architecture space exploration indicates up to 49% reduction in the total number of instructions executed through the introduction of a few custom instructions and small changes to the reference C source code. The accelerator is designed as an autonomous unit that can be attached to a configurable RISC CPU where it makes use of the host register file and Load/Store pipeline.

Introduction

Speech compression is utilized in a multitude of applications including amongst others VoIP networks and digital satellite systems. Typical consumer products comprise multimedia terminals, digital dictation machines, videophones and IP phones. The G723.1 recommendation [1] in particular was designed to standardize telephony and videoconferencing over public telephone lines (POTS) and is part of the ITU H.324 standard. ITU-T recommendations G723.1 and G729A [2] belong to the Code-Excited Linear-Prediction (CELP) category of speech coders.

This work describes architecture (instruction) extensions for the efficient execution of the above vocoders. These extensions are being implemented in a moderate-complexity datapath (coprocessor) attached to a configurable embedded processor.

Problem Formulation

To be able to utilize speech compression in a portable consumer product, it is essential to provide high-performance, low-power embedded DSP hardware. Previous work includes the progressive transformation of a GSM vocoder into VLSI hardware using the SpecC Methodology [7]. In [8], a custom VLIW-coprocessor is described to accelerate the half-rate GSM vocoder. In addition, there is a significant body of research into the automatic/semi-automatic targeted instruction set generation [3, 4, 5, 6]. Currently however, state-of-the-art configurable processor vendors follow a manual approach in extending their instruction sets. We take a similar approach since our hardware baseline is such a processor. We profile, measure and manually customize the ISA of a 32-bit MIPS-like processor such as to match precisely the requirements of our workload.

Experimental Method and Results

We utilized the SimpleScalar Toolset [9] to compile and simulate our workloads. Sim-profile was extended to

recognize the new instructions and the workloads where run with and without the new instructions switched on.

The basic_op.c package was modified such that each function mapped to hardware was replaced by three to five inline assembly statements. These functions were declared themselves as inline however, no performance difference was measured leading to the conclusion that the GCC 2.7.3 compiler automatically inlined them. Both workloads where compiled with full optimisations (-O7).

Figures 1 and 2 depict the normalized complexity of the G723.1 and G729A vocoders as a function of the extension instructions.

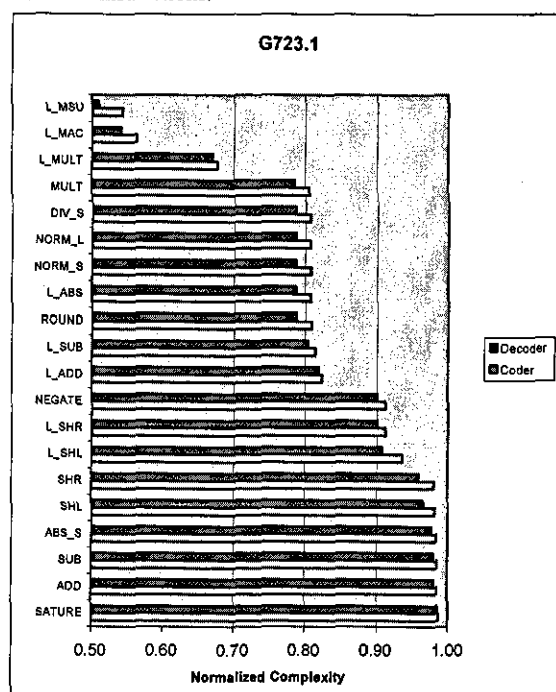


Figure 1: G723.1 Normalized Complexity

We observe that most benefit is achieved with the introduction of a pipelined multiply-shift (L_MULT), multiply-add (L_MAC), 32-bit add/subtract-saturate (L_ADD, L_SUB), 32-bit shifts with negative amount support (L_SHL and L_SHR), round and saturate instructions.

Figure 3 depicts a high-level view of a high-performance scalar processor (with limited dual-issue capability) incorporating the proposed datapath. Instructions are dispatched either to the integer pipeline or the extension datapath in the dispatch stage. The 16x16 multiplier is pipelined in order not to penalize the main processor cycle time. Single-cycle operators including all arithmetic, shifts

and saturation are handled during the Dcache2 stage where they are registered prior to being passed to the main pipeline. This is important since the setup time of the embedded RAM block (Register file) is typically two to three times longer than the setup time of an array of flops in a high-performance 0.13 um Silicon process thus creating a potential critical path in the Dcache2 stage.

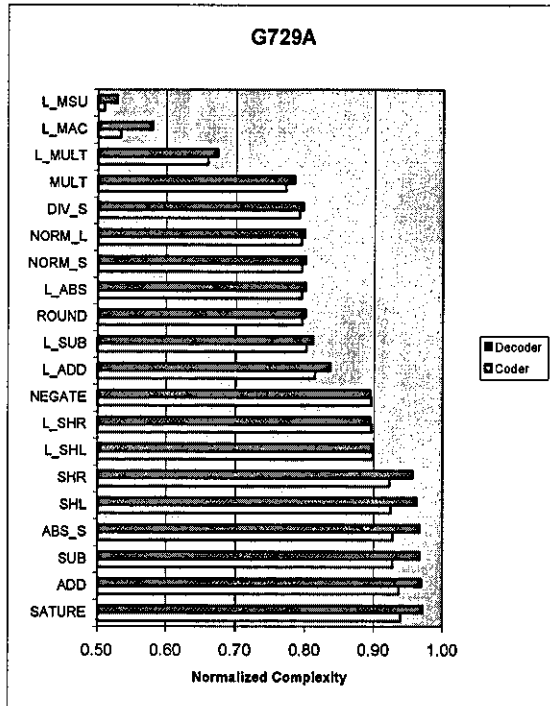


Figure 2: G729A Normalized Complexity

Conclusions

We have achieved close to 49% reduction in the algorithmic complexity of the ITU-T G723.1 and G729A speech coders. We are currently investigating a decoupled microarchitecture: Preliminary results indicate further significant reduction in the vocoder complexity through introducing a dedicated register file to the accelerator.

References

1. ITU-T Recommendation G.723.1, 'Dual Rate Speech coder for multimedia communications transmitting at 5.3 and 6.3 kbits/s', 3/96
2. ITU-T Recommendation G.729, 'Coding of speech at 8 kbits/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP)', 3/96
3. P. Faraboschi, G. Brown, J. Fisher, 'Lx: A technology platform for Customizable VLIW embedded computing', Proceedings of the 27th Annual International Symposium on Computer Architecture, Vancouver, Canada

4. Vinod Kathail, Shail Aditya, Robert Schreiber, B. Ramakrishna Rau, Darren C. Cronquist, Mukund Sivaraman, 'PICO: Automatically designing custom computers', IEEE Computer, 35(9), September 2002
5. M. Arnold, H. Corporaal, 'Designing Domain-specific processors', Proceedings of the 9th International Workshop on Hardware/Software Codesign, Copenhagen, April 2001
6. H. Choi Jong-Sun Kim, Chi-Won Yoon, In-Cheol Park, 'Synthesis of application specific instructions for embedded DSP software', IEEE Transactions on Computers, 48(6) 603-14, June 1999
7. A. Gerstlauer, S. Zhao, D. Gajski, 'Design of a GSM Vocoder using the SpecC methodology', TR ICS-99-11, University of California, Irvine
8. M. Prasad, P. Arcy, M. Diamondstein, H. Srinivas, 'Half-Rate GSM Vocoder Implementation on a Dual-Mac Digital Signal Processor', Proceedings of the 1997 IEEE International Conference on Acoustics, Speech and Signal Processing
9. D. Burger, T. Austin, 'Evaluating Future Microprocessors: The SimpleScalar Tool Set' <http://www.simplescalar.com>

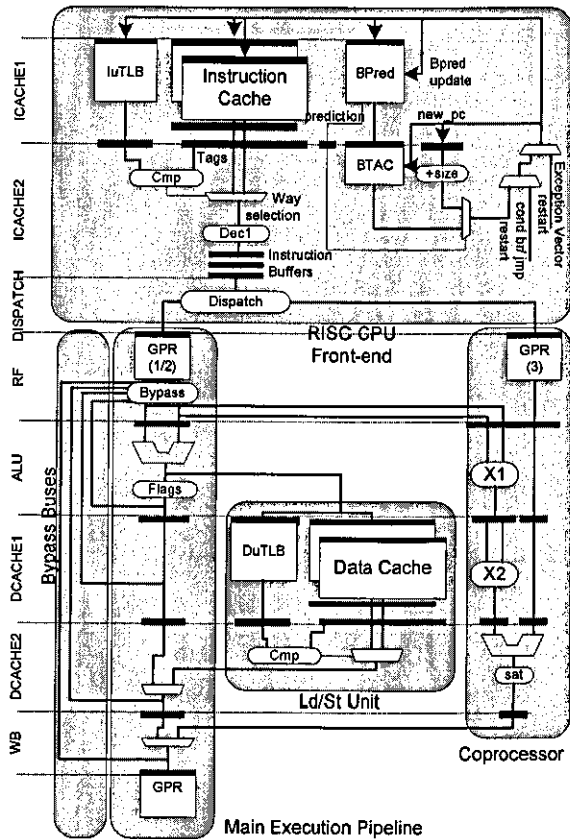


Figure 3: High-level processor pipeline

Paper PC2: V. A. Chouliaras, J. L. Nunez-Yanez, S. Agha, '*Silicon Implementation of a Parametric Vector Datapath for real-time MPEG2 encoding*', Proceedings of the IASTED (SIP) 2004, Honolulu, Hawaii, USA, ISBN: 0-88986-442-X

SILICON IMPLEMENTATION OF A PARAMETRIC VECTOR DATAPATH FOR REAL-TIME MPEG2 ENCODING

V. A. Chouliaras, J. L. Nunez-Yanez, S. Agha
Department of Electronic and Electrical Engineering
University of Loughborough
UK
v.a.chouliaras@lboro.ac.uk

ABSTRACT

We discuss the architecture specification, RTL development and ongoing physical implementation of a parametric vector/SIMD accelerator for real-time MPEG2 encoding. The MPEG2 TM5 reference code was systematically optimized through tapping the Data-Level-Parallelism (DLP) of the inner loop of Motion Estimation (ME) via custom vector extension instructions. We show that these instructions reduce the computational complexity of the encoding process by up to 60% for full-search motion estimation. The combined RISC CPU/Vector accelerator is being implemented as a hard macro. This work focuses on the flow from algorithmic C to a placed and routed database for the datapath of the vector accelerator in a high-performance 0.13 μm , 8-layer copper silicon process, for a number of register file configurations.

KEYWORDS

Video coding, multimedia processing, coprocessors, computer architecture

1. Introduction

The MPEG2 video coding algorithm is a very popular standard for lossy video compression used in many consumer products such as DVD players/recorders and digital set top boxes. The standard was introduced in 1994 by the ISO/ITU-T [1] organization to support good-quality video with transmission rates ranging from 4-80 Mbits/s. The MPEG2 codec is based around the discrete cosine transformation of either the residual data, obtained after performing motion estimation and compensation when removing redundancy within frames (inter-frame coding), or the original luminance/chrominance data when removing redundancy within the same frame (intra-frame coding). These transformations are followed by quantization which removes high spatial frequency components, significantly reducing the required transmission rate while maintaining good visual quality.

A significant amount of research is currently being conducted into alternative algorithms such as those based in wavelet transforms [2] and fractal-based coding algorithms [3]. Nevertheless, the DCT-based methods are presently much more popular than the other two and form

the basis of all international standards for digital video coding. These standards are summarized in Table 1.

Table 1: DCT-based video coding standards

Std	Year	Body	Rate	Usage
H261	1990	ITU-T	64 Kb/s	ISDN Video phone
MPEG1	1993	ISO	1.2 Mb/s	CD- ROM
MPEG2	1994	ISO/ITU-T	4-80 Mb/s	DVD, HDTV
H263	1995	ITU-T	64 Kb/s	PSTN Video Phone
MPEG4	2000	ISO	24-1024 Kb/s	Many
H264	2003	ISO/ITU-T	<64 Kb/S	Many

Two recently new video coding standards are the MPEG-4 [4] and H264 [5], introduced in 2000 and 2003 respectively. These algorithms achieve even lower bit-rates and higher PSNR values compared to MPEG2 through increasingly sophisticated techniques. This directly translates into significantly higher raw computational requirements (at least an order of magnitude increase for H264) and increased power consumption. This, in conjunction with the rising importance of digital video transmission (through the expected phasing out of analog TV over the next years and the popularity of video-capable embedded devices like mobile phones, portable DVD players) has spawned significant research and development efforts both in industry and academia into a new generation of sophisticated hardware platforms. These platforms utilize an increasing number of embedded configurable processors the most significant of which are reviewed in the next section.

2. Configurable and reconfigurable architectures

In an attempt to reach near-hardwired performance levels, embedded processor vendors have produced CPU architectures that can be extended to closely match the processing and memory requirements of the required

algorithm. This is the domain of (statically) configurable, extensible processors. It is interesting to note that traditional embedded CPU designers like ARM and MIPS have jumped in the configurable CPU bandwagon of pioneering companies like ARC and Tensilica via closely/loosely coupled coprocessors (ARM, MIPS) or datapath accessibility (MIPS). In the last few years, active research in the domain of very-long-instruction-word (VLIW) and dynamically configurable architectures has lead to the commercialization of more exotic architectures from vendors like SiliconHive, Aspx, Elixent and Cradle to name a few. The main vendors, architectures and characteristics of both configurable and reconfigurable offerings are summarized in Table 2:

Table 2: Configurable and Re-configurable processor vendors and architectures

Vendor	microarchitecture	Characteristics
ARC [6]	A4 (4 stages)	Scalar, 16/32-bit
	A5 (4 stages)	(modeless except A4),
	A600 (5 stages)	32-bit datapath,
	A700 (7 stages)	configurable, extensible
Tensilica [7]	Xtensa (5 stages)	Scalar, 16/24 bit
		(modeless), 32-bit datapath, configurable, extensible
ARM [8]	ARM9 (5 stages)	Scalar 16/32 bit
	ARM10 (6 stages)	(mode-bit), 32-bit
	ARM11 (8 stages)	datapath, coprocessor I/F
MIPS [9]	M4K (5 stages)	Scalar, 16/32-bit
		(mode bit), 32-bit datapath, coprocessor I/F, datapath extension technology
SiliconHive [10]	Avispa+	Instructions up to 768 bits long (ULIW). Up to 60 instructions per cycle
Aspx [11]	LineDancer	Combines a SIMD parallel processor with a RISC controller
Elixent [12]	DFA-1000	Consists of an array of 4-bit ALUs connected using a routing network of SRAM-based switches.
Craddle [13]	MDSP	Multiple RISC, DMA and DSP engines arranged in quads, in a two level hierarchical bus structure with local memories.

3. Architecture-level results

The MPEG2-TM5 reference software was initially profiled, in native mode (x86) as well as on a simulated

processor which implements the SimpleScalar ISA [14]. We used 12 video sequences (*fog, snowfall, snow lane, cup, deadline, office, paris, rotating city, student, mother and daughter, bowling, tennis*), each consisting of 25 frames. Profiling was done for full-search [15], three-step-search [16], four-step-search [17] and hierarchical-diamond-search [18]. As shown in Figure 1 (full-search) and Table 3 (algorithmic search methods), the major complexity contributors are the inner loop of the motion estimation function (DIST1) which computes the error of the current macroblock over an arbitrary reference macroblock. This function is called for all macroblocks in the search window of the reference frame and is independent of the search algorithm utilized.

For full-search ME in particular, our profiling results demonstrate that the DIST1 function complexity ranges on average from 51% to 72% of the unmodified reference software complexity for a search window of 6 to 62 pels respectively. At the same time, the complexity of FDCT and FULL_SEARCH varies with search window range with the former decreasing and the later increasing.

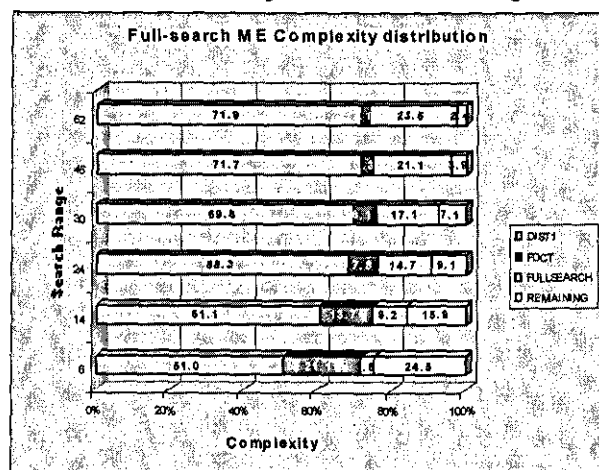


Figure 1: Full-search ME fractional complexity distribution

The algorithmic methods on the other hand exhibit near-constant behavior over the search range. Table 3 shows the average complexity distribution of the three identified functions for all remaining ME methods, averaged over all video sequences.

Table 3: Relative Complexity Distribution: Algorithmic ME

Function	Dist1	FDCT	Full Search	Others
3SS	43.55	18.09	1.55	36.80
4SS	42.23	18.57	1.04	38.16
HDS	41.54	18.57	1.93	37.95

It is therefore clear that the inner loop of the ME computation is the most processing-intense function and one that would provide the major performance benefit if accelerated successfully. The DIST1 function was subsequently recoded to expose the data-level-parallelism

and in the process, a vector ISA was identified. Figure 2 depicts the average complexity reduction for full-search ME, across all video sequences, over a search range of 6-62 pels and maximum vector lengths of 32, 64 and 128 bits. It demonstrates an increasing algorithmic complexity reduction with increasing search range due to the introduction of three vector instructions. These vector extensions are discussed in the next section.

We further observe that the difference in complexity reduction between the 32-bit (4 bytes) and 64-bit (8-bytes) and between 64-bits 128-bits (16-bytes) averaged over all sequences, ranges between 4.4% to 9.9% and 2.2% to 5 % respectively, over the search window range. This demonstrates that a datapath width of 64 bits (VLMAX=8) presents a good design compromise in terms of area-performance.

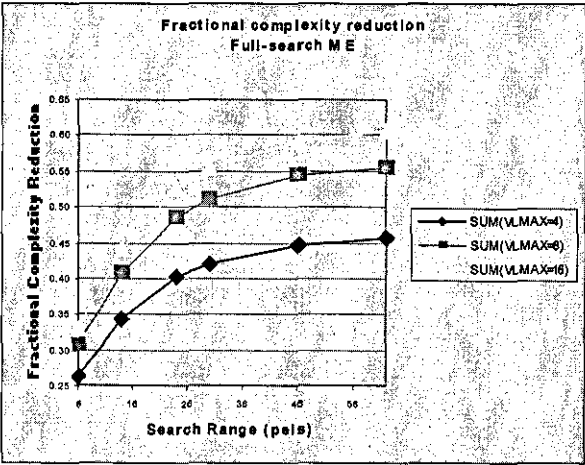


Figure 2: Average fractional Complexity reduction over all sequences vs. Search range, vs. VLMAX

4. Vector ISA and Programmers Model

This section discusses the programmer-visible part of the accelerator. Figure 3 shows the extra state added on top to the existing Sparc V8 processor state.

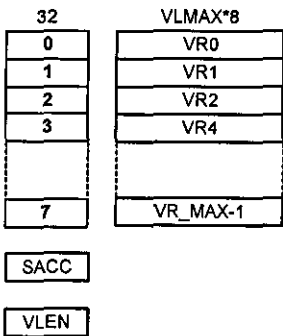


Figure 3: Accelerator Programmers Model

There are 8 32-bit scalar registers, used primarily for memory address calculation, a parametric scalar

accumulator used when executing the vector SAD instruction, the Vector length register VLEN which specifies the number of byte elements of the target vector register that will be affected by the currently executing vector instruction. Finally, there are up to VR_MAX, parametric-length vector registers (maximum length can be set to any value between 4 and 1024 bytes at elaboration time) that are used to hold the luminance data prior to executing the VSAD operation.

The coprocessor supports a number of vector operate, load/store and RISC communication instructions. These are detailed in [19] by the authors. Table 4 illustrates the vector compute instructions supported by the parametric datapath:

Table 4: Datapath Instructions

Command	Description
VSAD	Compute the absolute value of the difference of two 8-bit numbers. Accumulate result into scalar accumulator
VAVG2SAD	Average two 8-bit numbers and compute the absolute value of the difference of the average with a third 8-bit number. Accumulate result into scalar accumulator
VAVG4SAD	Average four 8-bit numbers and compute the absolute value of the difference of the average with a fifth 8-bit number. Accumulate result into scalar accumulator

5. Parameterization

A particular characteristic of our approach is the high-degree of parameterization starting from the algorithmic level, all the way down to the physical implementation. The parameterization constants belong to one of four different groups as shown in Table 5:

Table 5: System parameterization

Level	Parameters	Description
Algorithm	Maximum Vector length (VLMAX) Maximum number of vector registers (VR_MAX)	These parameters have a direct effect in the performance of the vectorized DIST1 function
RTL	Vector register file implementation (SRAM/Flop/Latch-based)	Affect the RTL implementation of the coprocessor. Includes the silicon area, power consumption and max operating frequency
RTL	Fast bypassing	Affects the timing of the RTL

implementation of the bypassing logic within the vector pipeline. Latency can be either 0 (default) or 1 (fast implementation)

Physical (Pre-route)	Floorplan aspect ratio, utilization	aspect	Affect the physical domain (routability, power consumption, area, IR-drop)
Physical (Route)	Power configuration Optimization effort	grid	Affect the final result of the router

At the highest level is algorithmic parameterization via VLMAX and VR_MAX. These affect the geometry of the vector register file and the number of scalar datapaths that constitute the vector pipeline. For the MPEG2 TM5 workload, a VR_MAX of 8 is sufficient for the stall-free implementation of the vectorized DIST1 function. Similarly, a VLMAX of 16 bytes (128 bits) and subsequently, a vector datapath consisting of 16 scalar elements, is the default parameter for the workload. Future algorithmic optimization will allow the use of VLMAX>16, further accelerating the workload.

At the RTL level, we can specify the implementation of the vector register file using dual-port SRAMs, flops or latches. In addition, a critical path in the result bypassing from the vector execute stage (EXEC) back to register fetch (DECODE stage) can be removed through the FAST_BYPASS parameter. This would move pipeline forwarding to EXEC2-DECODE resulting in 1-cycle latency across dependent vector-operate instructions. However, a maximum of 8 vector registers suffices to ensure that no dependent instructions appear back-to-back during the SAD computations in the degenerate (32-bit) case.

At the physical planning level (Physical pre-route), the user can specify minimum and maximum floorplan aspect ratios and target utilization and step values for each. A scripting mechanism controls the physical synthesis phase, iterating across the aspect ratio range, for all utilization values. The scripts collect the area/performance output from the synthesizer in a final results file which can be viewed in a spreadsheet. This is potentially the most time-consuming phase of the investigation and one that benefits the most from a multiprocessing/distributed processing.

The last level of parameterization directly affects the QoR of the routing phase. This includes the power grid configuration (metal layers, horizontal/vertical spacing) and the optimization effort for the router.

6. Coprocessor RTL Implementation

Figure 4 depicts the combined scalar processor/vector coprocessor in the context of a SoC sub-system. Our

microarchitecture utilizes the industry-standard AHB bus [20] to connect the streaming masters (Scalar CPU, DMA engine and Coprocessor LSU) to a single memory slave (PC133 SDRAM controller). The AHB datapath is 32-bits wide and clocked at the same rate as the rest of the system.

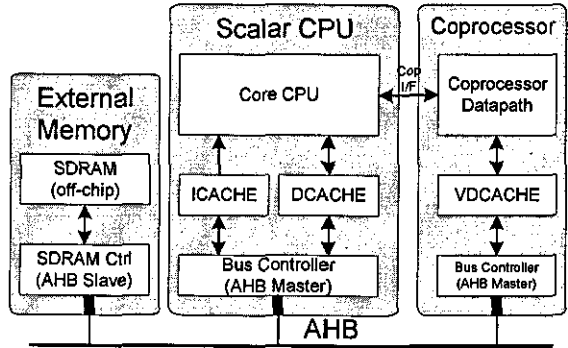


Figure 4: Coprocessor SoC Kernel

Figure 5 shows a detailed view of the microarchitecture of the combined processor/coprocessor indicating the bi-directional communication channel across the scalar CPU and the accelerator.

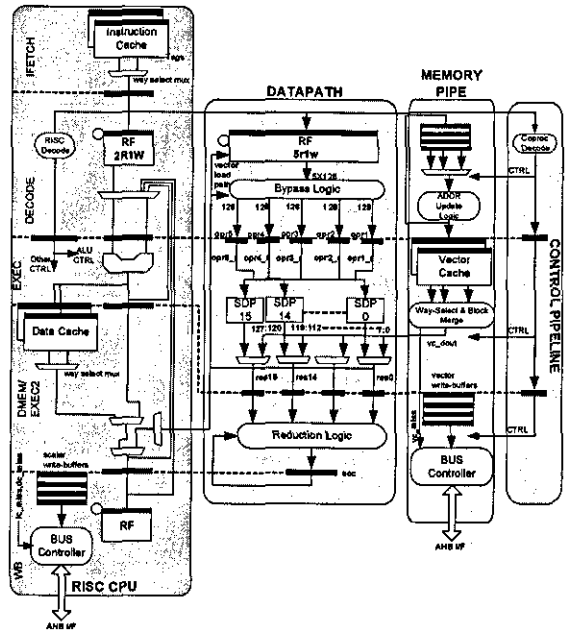


Figure 5: Accelerator Microarchitecture

The main CPU is a standard 5-stage RISC pipeline with the synchronous register file access taking place on the falling edge of the second stage (DECODE) due to architectural reasons [21]. A write-through data cache with three non-collapsing write buffers and AHB snooping ability is included and instructions are fetched from the parametric Instruction Cache.

The Coprocessor is segmented into three major sections: The datapath, the Memory Pipeline and the control Pipeline.

There is a bi-directional communication channel between the main RISC CPU and the vector accelerator. This is build on-top of the existing coprocessor I/F. Typical R/W transactions are depicted in Figure 6:

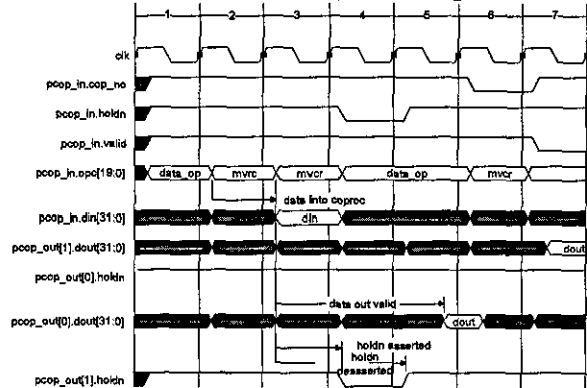


Figure 6: Coprocessor I/F Transactions

7. Vector Datapath Hard Macro

The combined processor-coprocessor architecture is currently being implemented as a hard macro, in the context of an AHB-based SoC Kernel for video coding acceleration, targeting a high-performance, 0.13 μm , 8-Cu silicon process. A highly-automated scripting methodology was used which, in conjunction to the design parameterization, can exhaustively probe the synthesis as well as the physical implementation space for each physical cluster of the processing kernel.

For this work we choose the parametric vector datapath. The scripting process iterated over tens of potential implementation candidates of the physical cluster and achieved a local minimum at an aspect ratio of 0.4-0.6 (width:height) and a pre-route utilization of 85%, for the RAM-based register file configuration. Subsequently, the aspect ratio was fixed at 0.5 for all the vector register file configurations. Figures 7, 8 and 9 depict the floorplan and layout of the vector datapath cluster for SRAM, Flop and Latch-based vector register file configurations respectively.

Our results indicate that very high post-route silicon utilization is achievable however, at the expense of operating frequency. Congested designs like the flop and latch-based versions achieved significantly better pre-layout (post-placement) timing to that depicted in the table above. The latch-based configuration in particular was penalized by the routing of a second, double-frequency clock used to generate the write-strobe. The critical path in all designs was inside the reduction logic and no register re-timing was attempted. The physical results are summarized in Table 6.

Table 6: Datapath physical macro data

Config	Fmax (MHz)	Util (%)	Std Cells + RAMS	Area (μm^2)
SRAM	312	93.02	26777+5	457x914 (417689)

Flops	278	88.28	59084+0	460x938 (431480)
Latch	229.7	91.76	46519+0	488x1003 (489464)

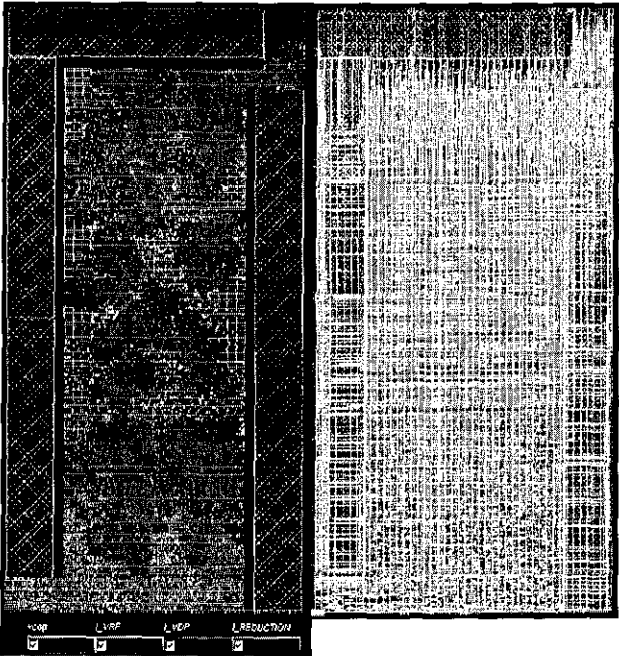


Figure 7: RAM-based VRF configuration

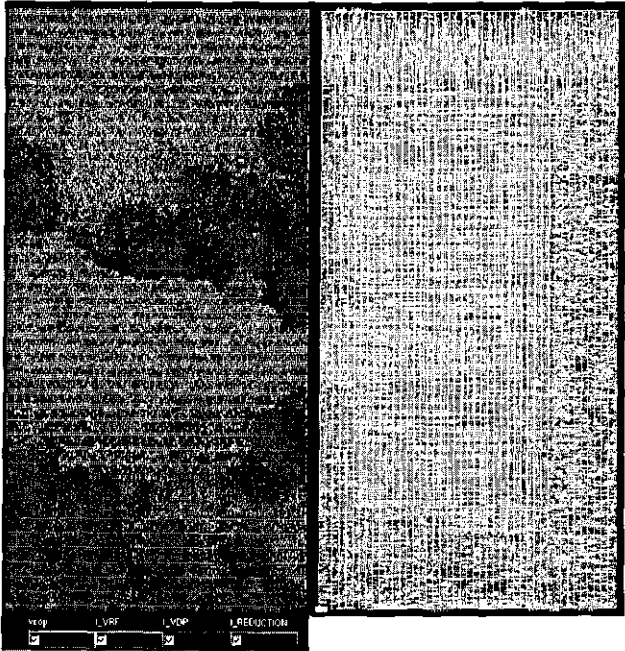


Figure 8: Flop-based VRF configuration

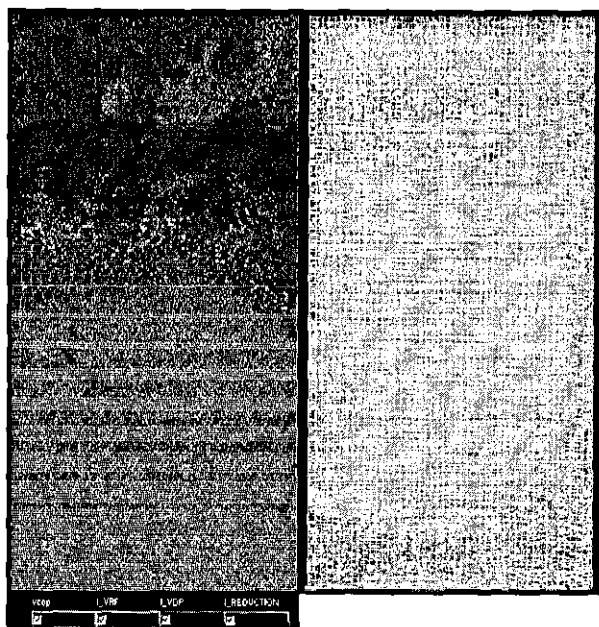


Figure 9: Latch-based VRF configuration

8. Conclusions and future work

This work discussed the architecture specification and physical implementation of a parametric vector datapath. Starting at the algorithmic level, we applied systematic modeling and transformation techniques to expose and evaluate the Data-Level-Parallelism of the inner loop of the Motion Estimation algorithm. In the process, we developed a custom, vector ISA which was implemented as a tightly-coupled coprocessor, attached to a RISC CPU. By targeting only the inner loop of the ME we are able to leverage our microarchitecture to accelerating a number of Algorithmic ME methods without modification. Through following a highly-parameterized design approach, we are able to specify top-level architecture, microarchitecture, timing and physical implementation constraints which are propagated down the implementation flow via a scripting mechanism. In this way, we are able to exhaustively probe the implementation space of our microarchitecture and converge to an appropriate physical solution. Ongoing work takes place in using this methodology at the SoC-kernel level where the described accelerator datapath, and its control and memory pipelines are connected to the controlling RISC CPU thus, forming a complete SoC computation kernel for real-time MPEG2 video encoding.

Architectural exploration results indicate further significant improvement through the vectorization of the standard, floating-point forward-DCT algorithm, at the expense of introducing floating-point operations in a consumer-SoC form-factor which however, contradicts the economics of consumer-SoC design. Additional research will therefore probe the architecture space for a number of integer-based forward DCT algorithms and their

vectorization benefit. We expect to utilize much of our existing hardware infrastructure in this process. A further software-based optimization comes in the form of spatial optimization (re-arrangement) of the luminance data in order to accommodate wider (>16 bytes) vectors. It is expected that the increase in processing complexity required to restructure the luminance arrays will be amortized over greater vector lengths.

Finally, the last and potentially very significant source of parallelism in block-based video coding algorithms is Thread-Level Parallelism. In this case, multiple processor contexts execute different sub-graphs of the control-flow graph of the algorithm while maintaining sequential semantics through Fork/Join operation. Our results indicate that static-threading of the Full-search motion estimation and Forward DCT computations provides very significant extra benefit which will complement the Data-Level-Parallelism optimizations presented in this work. The Electronics Systems Design Group at Loughborough University actively pursues this route.

References

- [1] S. Liu, 'Performance comparison of MPEG1 and MPEG2 video compression standards', 41st IEEE International Computer Conference COMPCON 96, pp. 25-28, 1996.
- [2] P. Orbaek, 'A real-time software video codec based on wavelets', Proc. Of Intl. Conf. On Communication Technology (IFIP), 2000.
- [3] J. Streit, L. Hanzo, 'A Fractal Video Communicator', IEEE Vehicular Technology Conference (VTC), pp. 1030-1034, Stockholm, Sweden, 1994.
- [4] S. Vassiliadis, G. Kuzmanov, S. Wong, 'MPEG-4 and the New Multimedia Architectural Challenges', Proc. 15th International Conference on Systems for Automation of Engineering and Research (SAER-2001), pp. 24-31, Bulgaria, 2001.
- [5] 'Emerging H.26L Standard: Overview and TMS320C64x Digital Media Platform Implementation', White Paper, UB Video Inc., Vancouver, Canada, 2002.
- [6] <http://www.arc.com/products/SOC/microprocessors/arcprocessors/index.html>
- [7] <http://www.tensilica.com/html/configurability.html>
- [8] <http://www.arm.com/products/CPUs/embedded.html>
- [9] <http://www.mips.com/content/Products/Cores/32-BitCores>
- [10] www.siliconhive.com
- [11] www.aspx-semi.com
- [12] www.elixent.com
- [13] www.cradle.com
- [14] D. Burger, T. Austin, 'Evaluating Future Microprocessors: The SimpleScalar Tool Set', <http://www.simplescalar.com>
- [15] <http://www.mpeg.org/MPEG/MSSG/>
- [16] Zeng and Liu 'A new 3 step search Algorithm for Block Motion Estimation', IEEE Transactions on Circuits and Systems for Video Technology, Vol. 4, No 4, Aug. 1994.
- [17] Lai-man Po and Wing-Chung Ma, 'A novel four step-search algorithm for fast block motion estimation', IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, pp. 313-317, 1996.
- [18] J.Y.Tham, S.Ranganath, M.Ranganath and A.A.Kassim, 'A novel unrestricted center-biased diamond search algorithm for block motion estimation', IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, pp 369-377, 1998
- [19] V. A. Choularas, J. L. Nunez, 'From C to Si: Codesign of a parametric embedded vector coprocessor for high-performance MPEG2 video encoding', submitted to IEEE Transactions on Computers
- [20] 'AMBA Specification (Rev 2.0)', www.arm.com
- [21] 'The Sparc Architecture Manual Version 8', www.sparc.com

Paper PC3: V. A. Chouliaras, J. L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso '*A multi-standard video coding accelerator based on a vector architecture*', Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2005), Las Vegas, Nevada, USA, ISBN: 07803-8839-9

A multi-standard video coding accelerator based on a vector architecture

Vassilios A. Chouliaras, Jose L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso

Abstract — We discuss the architecture definition and microarchitecture of a multi-standard, parametric vector accelerator for block-based video coding. Our target coding algorithms were the MPEG-2 TM5, MPEG-4 (XViD) and STM's proprietary H.264 implementation. We fully vectorized the MPEG-2 and MPEG-4 coders and partially vectorized the H.264 encoder. In the proprietary H.264 case, we targeted inner loop of the motion estimation function. Our preliminary results demonstrate a significant complexity reduction of the order of 65%, 70% and 16% for MPEG-2, MPEG-4 and H.264 respectively. In the latter case the complexity of the inner loop of motion estimation has been reduced by 79% compared to the scalar case.¹

Index Terms — RISC Coprocessor, SIMD, MPEG2, MPEG4, H264.

I. INTRODUCTION

Vector/SIMD architectures are the most effective means for tapping the abundant data-level parallelism present in current and emerging embedded workloads [1, 2] with ever-increasing transistor budgets permitting the use of complete, short-vector units within desktop processors [3] as well as high-volume, SoC-based consumer products. [4].

In this work, we define the Instruction Set Architecture (ISA) and microarchitecture of a parametric vector coprocessor capable of significantly accelerating the three most important video coding algorithms namely MPEG-2 [5], MPEG-4 [6] and the evolving H.264 standard [7]. The vector accelerator is tightly-coupled to an open-source, configurable extensible Sparc-V8 compliant RISC processor [8]

The novelty of our work lies with identifying a number of common vector instructions that afford significant performance benefit in the three major video coding standards of interest and their subsequent implementation in the form of a custom vector accelerator [9]. The baseline architecture for our investigation is a MIPS-like machine [10].

II. METHODOLOGY AND RESULTS

The workloads were initially profiled and the complexity distribution recorded at function-level granularity. Fig. 1,

depicts the complexity distribution for MPEG-2 TM5, MPEG-4 (XViD) and the relative complexity of the Motion Estimation function in the proprietary H.264, which exploits a fast predictive-recursive algorithm, called Openslim, being able to achieve nearly the same quality performance of the Full-Search Block-Matching with only a few percent of the computation [11].

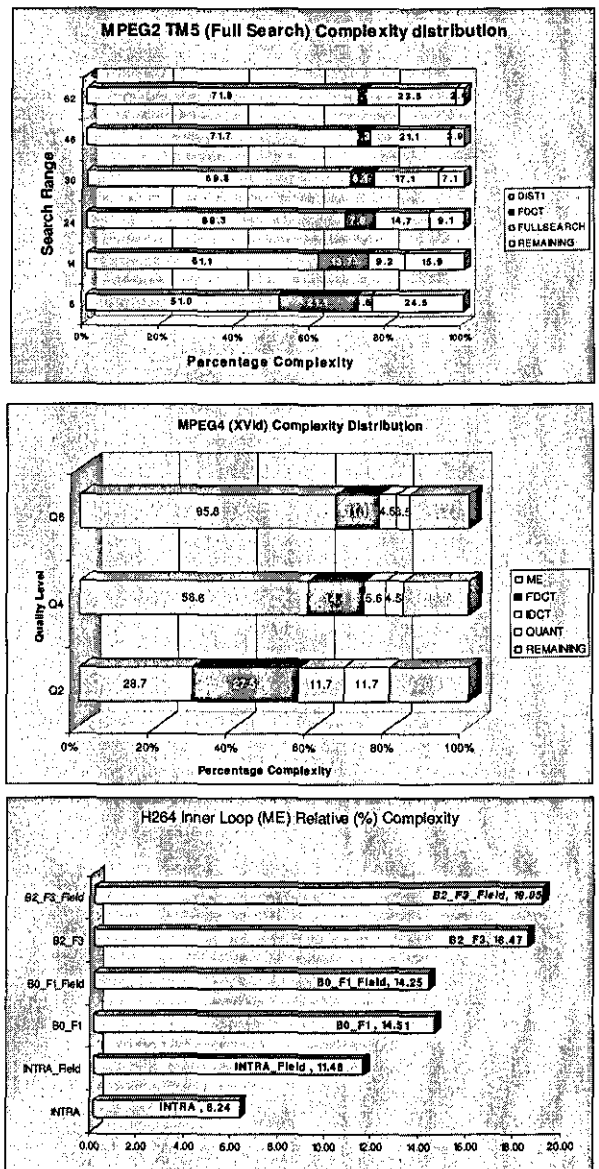


Fig 1: Video Coding Standards Profiling

¹ V. A. Chouliaras is with the Department of Electronic and Electrical Engineering, University of Loughborough, UK

Jose. L. Nunez is with the Department of Electronic Engineering, University of Bristol, UK

F. S. Rovati and D. Alfonso are with ST Microelectronics, Advanced System Technology Group, Agrate, Italy

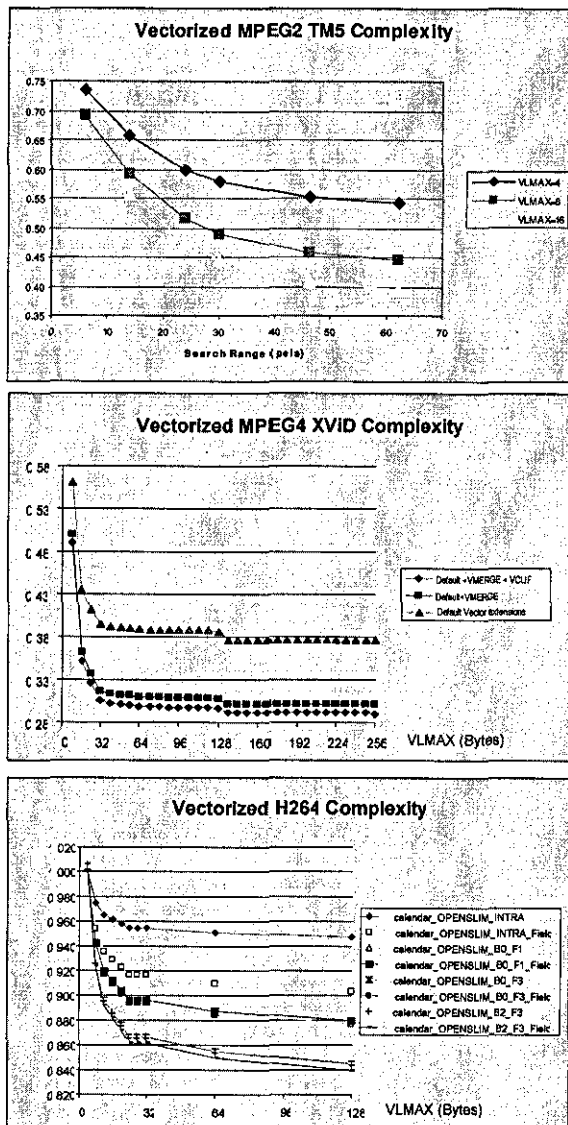


Fig. 2: Vectorized Video Coder Complexity

Fig 2 depicts the normalized (%) complexity of all three the vectorized video coders. The H.264 results are preliminary, with vectorization applied to the ENC_SATD function only (up to 72% complexity reduction in ENC_SATD). Further significant benefits are expected as other functions are vectorized.

III. MICROARCHITECTURE

The vector extensions are implemented as a tightly-coupled coprocessor, attached to a configurable, 32-bit Sparc V8-compliant processor. The combined processor/coprocessor microarchitecture is depicted in Fig. 7. In the diagram, the main RISC processor supplies instructions to the vector pipeline during the decode stage. The vector register file is then accessed followed by operand bypassing in both pipelines. The resolved vector operands are then clocked into the operand registers. During EXEC1, all cross-lane

operations are performed (including permute, pack/unpack etc) as well as the first stage of the remaining operations. Intermediate results are pipelined to the next stage for the final stage of vector execution. For SAD (reduction) operations, the custom SAD datapath is used. Results commit to a staging register prior to being written to the vector register file on the falling edge of the clock

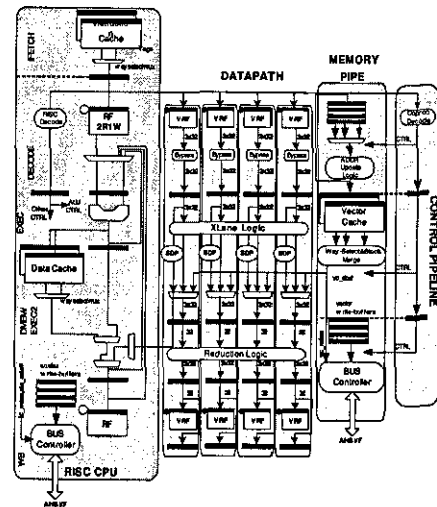


Fig. 3: Accelerator microarchitecture

IV. CONCLUSIONS

We developed custom vector instructions in the form of a tightly-coupled vector accelerator, to significantly reduce the complexity of the existing and emerging video coding standards. Further investigation will focus on Thread-level parallelism. Preliminary data on a thread-parallel MPEG-2 implementation reveal further, significant complexity reduction for 16 processor contexts.

REFERENCES

- [1] K. Asanovic, "Vector Microprocessors", Ph.D. Thesis, Technical Report UCB/CSD-98-1014, Computer Science Division, University of California at Berkeley
- [2] A. D. Paterson et al, 'A case for intelligent RAM: IRAM', IEEE Micro, April 1997
- [3] Peleg, U. Weiser, 'MMX Technology to the Intel Architecture', IEEE Micro, July 1996
- [4] K. Diefendorff, 'Sony's Emotionally Charged Chip', Microprocessor Report, vol. 13, no. 5, April 19 1999
- [5] <http://www.mpeg.org>
- [6] <http://www.xvid.org>
- [7] R. Schafer, T. Wiegand, H. Schwarz, 'The Emerging H.264 standard', EBU Technical Review, January 2003
- [8] 'The Leon-2 processor User's manual, XST edition, ver. 1.0.14', <http://www.gaisler.com>
- [9] V. A. Chouliaras, J. L. Nunez-Yanez, S. Agha, 'Silicon Implementation of a Parametric Vector Datapath for real-time MPEG2 encoding', to appear in IASTED 2004 (SIP), Honolulu, Hawaii, USA
- [10] D. Burger, T. Austin, 'Evaluating Future Microprocessors: The SimpleScalar Tool Set', <http://www.simplescalar.com>
- [11] D. Alfonso, D. Bagni, L. Celetto, S. Milani, 'Constant bit-rate control efficiency with fast motion estimation in H.264/AVC video coding standard', to be published in Proceedings of the 12th European Signal Processing Conference (EUSIPCO) 2004, Wien, Austria.

Paper PC4: V. A. Chouliaras, J. A. Flint, Y. Li, '*Parametric Data-Parallel architectures for TLM acceleration*', Proceedings of the 3rd International Conference on Computational Electromagnetics and Its Applications (ICCEA), Nov. 1-4 2004, Beijing, China

Parametric Data-Parallel Architectures for TLM acceleration

V. A. Chouliaras, J. A. Flint, Y. Li,
Department of Electronic and Electrical Engineering
University of Loughborough, UK

Postal address : Mr Vassilios A. Chouliaras, Dept of Electronic and Electrical Engineering,
University of Loughborough, Loughborough, LEICS LE11
3TU, UK

Email: v.a.chouliaras@lboro.ac.uk

Website: <http://www.lboro.ac.uk/departments/el/research/esd/projects.html>

Postal address : Dr. James A. Flint, Dept of Electronic and Electrical Engineering, University of
Loughborough, Loughborough, LEICS LE11 3TU, UK

Email: j.a.flint@lboro.ac.uk

Website: <http://www.lboro.ac.uk/departments/el/staff/flint-james.html>

Postal address : Mr. Yibin Li, Dept of Electronic and Electrical Engineering, University of
Loughborough, Loughborough, LEICS LE11 3TU, UK

Email: Y.Li2@lboro.ac.uk

Website: http://www.lboro.ac.uk/departments/el/research/esd/vpd_area.html

Parametric Data-Parallel Architectures for TLM acceleration

V. A. Chouliaras, J. A. Flint, Y. Li,
Department of Electronic and Electrical Engineering
University of Loughborough, UK

Abstract: We discuss the architecture and microarchitecture of a scalable, parametric vector accelerator for the TLM algorithm. Architecture-level experimentation demonstrates an order of magnitude complexity reduction for vector lengths of 16 32-bit single-precision elements. We envisage the proposed architecture replicated in a SoC environment thus, forming a multiprocessor system capable of tapping parallelism at the thread level as well as the data level.

INTRODUCTION

Prior attempts to implement the TLM algorithm [1] on general-purpose architectures have fallen into two major categories: Shared memory, cache coherent multi-processors [2, 3] and distributed processors [4] with shared-memory machines often demonstrating better performance.

The TLM is a highly-parallel three-dimensional numerical algorithm which has the potential for being accelerated along its innermost loop via vectorization thus, tapping parallelism at the data level (DLP). Furthermore, the algorithm can be statically 'sliced' (threaded) along the second outer loop, and be executed on the previously mentioned platforms via different processors executing different iterations. Such parallelism is known as thread-level-parallelism [5] and is currently being pursued by all major microprocessor vendors.

Successful acceleration of such parallel codes depends very much on the algorithmic communication pattern which dictates the level of data sharing across the multiple processors. In the case of the TLM, data transfers between individual nodes is very high and in extreme cases the data transfer during the *connect* part of the algorithm can be much more computationally expensive than the numerical calculations during *scattering*. The performance differential between shared memory and distributed machines is often attributed to such data sharing issues.

Custom architectures for accelerating TLM codes have been proposed in the past by Stothard and Pomeroy [6]. Our work proposes a custom vector approach to accelerating the inner loop of TLM

codes, quite unlike this earlier work. In our case, an embedded 32-bit processor is augmented with a configurable, extensible custom vector accelerator and resides on an on-chip-bus [7] thus, forming a finely-tuned SoC computation kernel for the TLM algorithm.

VECTOR ARCHITECTURE

The programmer's model of the parametric vector accelerator for TLM is depicted in Figure 1:

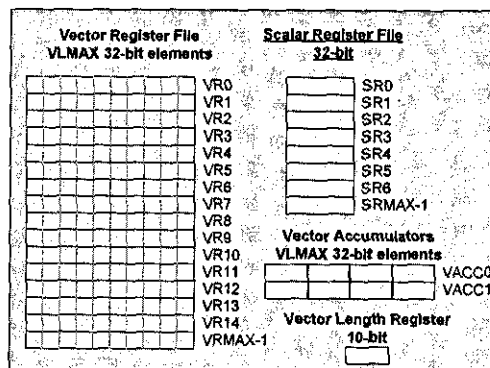


Figure 1: Vector accelerator programmer's model

The programmer's model specifies a parametric number of vector registers (VRMAX), each consisting of a parametric number of 32-bit single-precision elements (VLMAX). There is a scalar register file consisting of a parametric number of scalar 32-bit elements (SRMAX), used for virtual address computation, immediate passing and vector splat operations. Additionally, there are two vector accumulators each holding VLMAX single-precision elements and finally, the vector length register (VLEN) which specifies the number of bytes that will be affected by the currently executing vector opcode. The Instruction Set Architecture (ISA) of the accelerator includes standard vector floating point operations except division, vector Load/Stores, and a generalized permute instruction. A large number of sub-element manipulation instructions (including vector splat instructions) can be synthesized based on

the three-operand permute infrastructure. The ISA is summarized in Table 1

Table 1: Vector Coprocessor ISA

Instruction	Description
MVSR2VLEN	Transfer scalar register to vector length register (VLEN)
MVSR2CSR	Transfer RISC scalar register to coprocessor scalar register
MVCSR2R	Transfer coprocessor scalar register to RISC register
MVSR2CVEL	Move RISC scalar register to coprocessor vector element
MVCVEL2R	Move coprocessor vector element to RISC scalar register
VLDU	Load vector register unaligned under VLEN
VSTU	Store vector register unaligned under VLEN
VPERM	Three-operand bitwise vector permute
VSPLAT	Splat coprocessor scalar register to coprocessor vector register
VFPADD.S	Vector floating-point add (single precision) under VLEN
VFPSUB.S	Vector floating-point sub (single precision) under VLEN
VFPMUL.S	Vector floating-point mult (single precision) under VLEN
VFPMAC.S	Vector floating-point multiply-accumulate

VECTOR MICROARCHITECTURE

The proposed vector extensions are implemented as a tightly-coupled vector accelerator attached to an open-source, configurable, extensible, Sparc V8-compliant RISC CPU [8]. The processor/coprocessor combination communicates via the AHB On-Chip Bus to the SDRAM controller which controls the off-chip SDRAM part. A high level schematic of the scalar processor and vector accelerator is depicted in Figure 2.

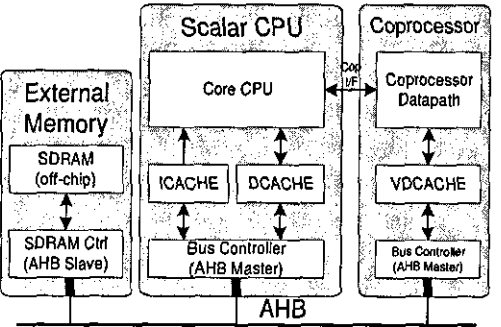


Figure 2: TLM computation kernel

As shown in the figure, there exists a bidirectional communication channel across the scalar processor and the vector accelerator. Though the open source CPU provides a coprocessor interface, it was decided to implement that channel in order to ensure pipelined, lockstep operation of the accelerator and timely transfer of data to and from the main CPU. Typical transactions on the developed channel are depicted in Figure 3.

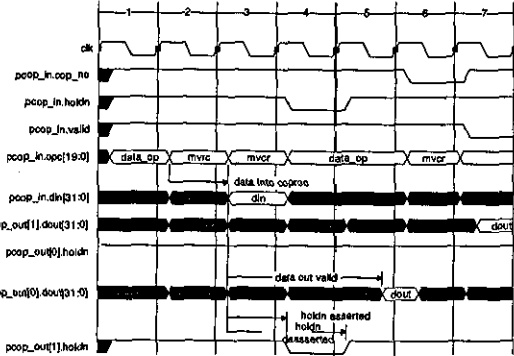


Figure 3: Processor-Coprocessor communication channel

The detailed microarchitecture of the combined scalar processor/vector coprocessor for a vector length of two 32-bit (single-precision) elements is depicted in Figure 4.

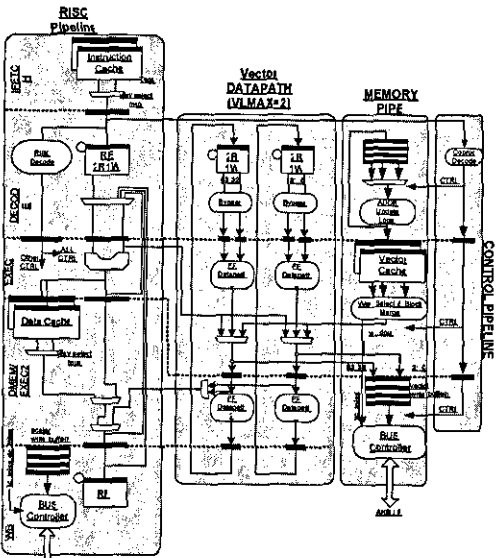


Figure 4: Detailed microarchitecture

Instructions are fetched from the multi-way set-associative instruction cache and stored in a single 32-bit register. Typically, high-performance RISC processors of equal pipeline depth would extract the source operand fields right after instruction cache access and set up the synchronous register file.

Unfortunately, that is not the case in the particular processor which, due to the windowing scheme of the Sparc V8 architecture, requires access to the current-window-pointer (CWP) register in order to compute a physical register file address. As a result, source operand addresses are set-up on the falling edge of the clock in the DECODE stage. During this stage, the register file is accessed and the two source registers are retrieved. Operand bypassing takes then place and the resolved operands are clocked into the ALU input registers, ready for execution. It is during this stage that the vector opcodes are identified and dispatched to the tightly-coupled vector accelerator. Decoding logic in the later produces a number of control fields which are pipelined down the control pipeline. Vector operand accesses are triggered by the falling edge of the clock during decode, for reasons of symmetry to the scalar pipeline. During the EXEC stage, the RISC CPU executes the scalar instruction or computes the virtual address of a Load/Store operation. In the same stage, the vector accelerator performs the first stage of the pipelined floating point computations. In the next stage, scalar data return to the main processor via the data cache return path whereas the vector accelerator performs the last stage of execution. Due to the very tight timing constraints, floating point results are stored in an intermediate register prior to committing to the vector register file.

METHODOLOGY

We have applied a basic implementation of the SCN TLM algorithm [1] in which no external boundary conditions were used. In the particular case, a single output node was used as a diagnostic aid to verify correct operation. We used the accelerated scatter method of Naylor and Ait-Sadi as proposed in [9]. The non-vectorized (scalar) algorithm was profiled both in native mode (IA32 Linux) as well as on our simulated processor for consistency of results. Scalar code profiling revealed a scatter:connect complexity ratio of 63:37, averaging over all the studied configurations.

Our simulation infrastructure is based around the simpliscalar toolset [10] which provides a complete computer architecture modelling and performance evaluation environment. The compiler used was GCC 2.7.3 with optimizations (-O3).

RESULTS

The reference problem chosen for benchmarking was a fixed mesh of 10^6 nodes. This number is convenient as it gives a prime factorisation of $2^6 \times 5^6$, which allows for the aspect ratio of the problem space to be varied over a reasonable range whilst maintaining the same number of nodes.

We measured the absolute complexity (dynamic instruction count) of the scalar code for all configurations of interest. Then, the vectorized code was run and its complexity recorded for a maximum vector length of up to 16 single-precision elements. Figures 2 and 3 depict the normalized complexity of the vectorized algorithm over maximum vector length.

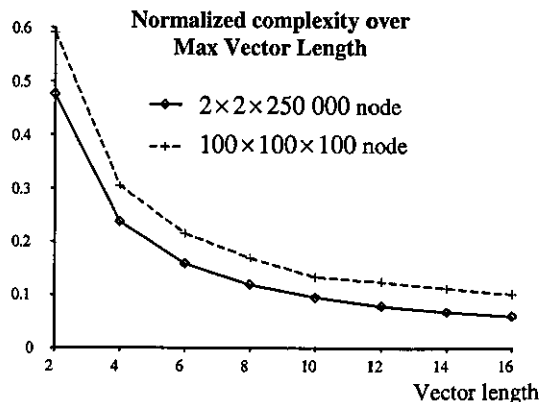


Figure 2 – Benchmarking using a thin and a cubic problem space.

Figure 2 suggests that the optimal (less complex) configuration is where the problem space is thin, i.e. where the vector length is maximised.

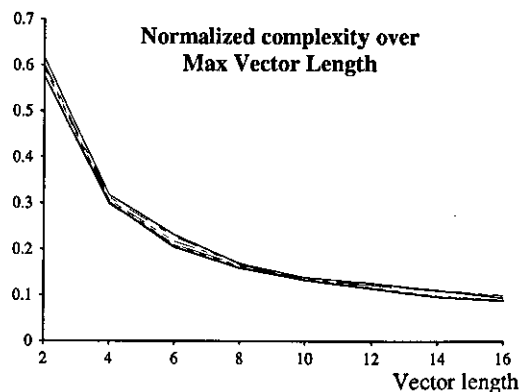


Figure 3 – Iteration time for $80 \times 100 \times 125$ node mesh with differing alignment relative to the vector direction. All of these results show a similar speedup

Figure 3 depicts a $80 \times 100 \times 125$ configuration compared with a mesh of $100 \times 125 \times 80$, $80 \times 125 \times 100$, etc. These mesh dimensions were chosen as being typical of realistic model of an electromagnetic scattering situation. Results demonstrate that vectorization alignment changes only slightly the complexity (and hence run time) in all configurations. A vector length of 16 single-precision elements showed a speedup of

approximately an order of magnitude thus clearly demonstrating the benefit of using parallelism at the data level.

CONCLUSIONS

We have proposed a parametric vector accelerator to exploit the significant amount of data level parallelism which is inherent within the TLM code. Our results demonstrate an order-of-magnitude performance improvement can be achieved for a vector length of 16 single-precision elements. Such a configuration is realizable with current VLSI technology.

We are also actively investigating thread-level parallelism as the second major source of parallelism in the workload. Our scalable architecture can be replicated thus, creating a cache-coherent, embedded multiprocessor for TLM acceleration providing further performance benefits.

REFERENCES

1. P. B. Johns, "A symmetrical condensed node for the TLM method", IEEE Trans. Microw. Theory Tech., vol. 35, no. 4, pp. 370-377, 1987.
2. J. L. Dubard, O. Benevello, D. Pompei, J. Le Roux, P. P. M. So, and W. J. R. Hofer, "Acceleration of TLM through signal processing and parallel computing", in *Computation in Electromagnetics*, pp. 71-74, IEE, 25-27 November 1991.
3. C. C. Tan and V. F. Fusco, "TLM modelling using an SIMD computer", Int. J. Numerical Modelling: Electronic Networks, Devices and Fields, vol. 6, pp. 299-304, 1993.
4. P. J. Parsons, S. R. Jaques, S. H. Pulko, and F. A. Rabhi, "TLM modeling using distributed computing", IEEE Microw. and Guided Wave Lett., vol. 6, no. 3, pp. 141-142, 1996.
5. J. Henessy, D. A. Patterson "Computer architecture: A quantitative approach", Morgan Kaufmann publishers, ISBN 1-55860-329-8
6. D. Stothard and S. C. Pomeroy, "Dedicated TLM array processor", Applied Computational Electromagn. Soc. J., vol. 13, no. 2, pp. 188-196, 1998.
7. "AMBA Specification (Rev 2.0)", www.arm.com
8. "The Leon-2 processor User's manual, XST edition, ver. 1.0.14", <http://www.gaisler.com>
9. P. Naylor and R. Ait-Sadi, "Simple method for determining 3-D TLM nodal scattering in nonscalar problems", Electron. Lett., vol. 28, no. 25, pp. 2353-2354, 1992.
10. D. Burger, T. Austin, 'Evaluating Future Microprocessors: The SimpleScalar Tool Set', <http://www.simplescalar.com>

Paper PC5: V. A. Chouliaras, J. L. Nunez-Yanez, T. R. Jacobs and Ashwin K. Kumaraswamy, '*Configurable Multiprocessors for high-performance MPEG-4 video coding*', Proceedings of the IEEE Annual Symposium on VLSI, May 11-12 2005, Tampa, Florida, USA

Configurable Multiprocessors for high-performance MPEG-4 video coding

V. A. Chouliaras, T. R. Jacobs and Ashwin K. Kumaraswamy

Loughborough University, UK

Jose L. Nunez-Yanez

University of Bristol, UK

E-mail: v.a.chouliaras@lboro.ac.uk

Abstract

We investigate the performance improvement of a multithreaded MPEG-4 video encoder executing on a configurable, extensible, SoC multiprocessor. Architecture-level results indicate a significant reduction in the dynamic instruction count of the order of 83% for 16 processor contexts compared to the original single-thread implementation. We extended an open-source 32-bit RISC CPU to include hardware-based multi-processing primitives and associated support state and implemented a parametric, bus-based SoC multiprocessor as the target platform for the threaded video encoder.

1. Introduction

The past 10 years have seen a substantial increase in the quantity of audio-visual information (multimedia content) that must be processed and delivered to consumers. This has initiated a large body of research, both in industry as well as in academia, resulting in advanced video coding standards such as MPEG-2 [1], MPEG-4 [2] and H264 [3]. These standards enabled major reductions in the channel bit-rates via advanced compression algorithms exploiting redundancy in the spatial (intra-frame) and temporal (inter-frame) dimensions of the input video sequence. A common characteristic of all three standards is the very high computational requirements of the encoding process [4]. For the MPEG-4 in particular, our data indicate that 400 MIPS are required to achieve 30 Frame-per-second (FPS) at QCIF resolution (176x144). Such processing requirements are unrealistic, from a business perspective, in a mobile, wireless consumer platform calling for the introduction of advanced optimizations in the encoding process and new VLSI architectures, capable of extracting all parallelism out of a block-based video coder workload.

Prior work by our group as well as other researchers has focused on Data-Level-Parallelism (DLP) since this is the most widely accepted form of parallelism in

media workloads. This work investigates the effect of exploiting the inherent Thread-Level-Parallelism (TLP) of block-based video coding standards as an orthogonal (thus, complementary) form of parallelism in trying to achieve high-performance, software-only solutions to the very high computational requirements of the encoding process.

2. Simulation Methodology

We developed a novel, multi-context, instruction-set-simulator (ISS) based on the SimpleScalar toolset [5]. The simulator is parametrical as to the number of processor contexts (software threads) it supports thus, permitting the theoretical study of arbitrary parallel configurations including shared-memory multiprocessors, multi-threaded processors and multithreaded multiprocessors. Architectural hooks are in place to allow it to interface to a cycle-accurate back-end thus permitting measurements of cycle effects such as clocks-per-instruction (CPI) per processor and interconnect bandwidth utilization. The simulator executes every instruction in 1 'time period' thus, can be classified as a PRAM model.

3. Results

The performance of the multi-threaded MPEG-4 encoder was evaluated with three video sequences, 'Garden', 'Foreman' and 'Coastguard'.

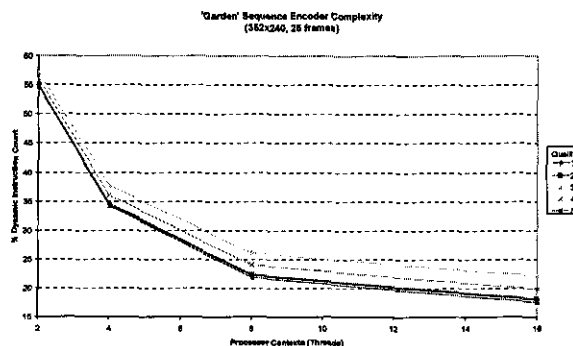


Figure 1: Garden Sequence Results

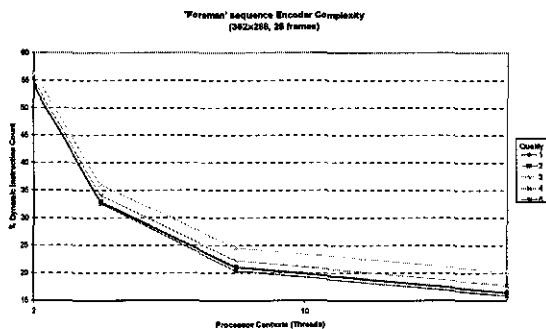


Figure 2: 'Foreman' sequence results

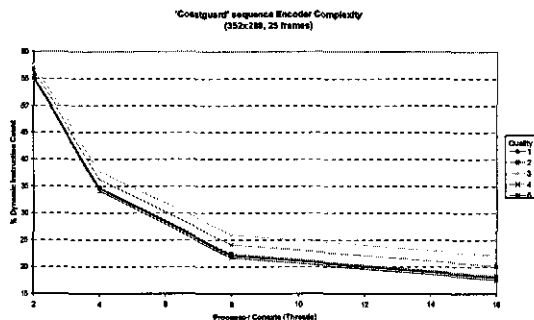


Figure 3: 'Coastguard' sequence results

All sequences were coded at a resolution of 352x240 pels with the exception of 'Foreman' which was coded at 352x288 pels. All sequences consisted of 25 frames.

Figs. 3, 4 and 5 depict the complexity metric as a function of processor contexts and quality metric. The graphs demonstrate an average 83% complexity-metric reduction at 16 processors, whereas a 2-processor configuration shows a near-linear complexity-metric reduction of the order of 45%. The results are collected on our PRAM simulator and include synchronization overheads however, do not model cycle effects such as the CPI ratio increase expected in a bus-based SoC multiprocessor.

4. VLSI Macrocell

We implemented the N=2 configuration of the system depicted in Fig. 6b in a high performance 0.13 μm CMOS process. The design was synthesized for maximum performance initially on Synopsys *Design Compiler* and then, read into Cadence *SoC Encounter* where floorplanning and power routing took place. The clusters were exported to Synopsys *Physical Compiler* for placement optimization and imported again into SoC encounter for detailed routing. Figs. 7 and 8 depict the floorplan and final layout of the N=2 MP configuration. The macrocell implementation data are tabulated in Table 1.

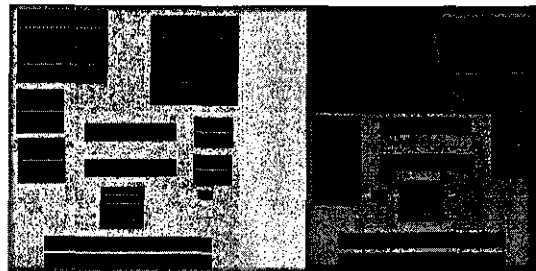


Figure 4: 2-way SMP floorplan



Figure 8: 2-way VSMP layout

Table 1: 2-way SMP VLSI macrocell data

PARAMETER	VALUE
Std cells	110099
RAMs	52
Fmax	179.5 MHz
Size	4585x2291 μm^2 (9086988 μm^2)

5. Conclusions

We discussed the development of a multi-threaded video encoder based on an open-source implementation of the MPEG-4 standard for a SoC Multiprocessor. Architecture-level experimentation showed a significant reduction of the order of 83% in the dynamic instruction count metric of the threaded algorithm compared to the original, sequential version clearly demonstrating the potential of exploiting the inherent TLP of video coding workloads.

6. References

1. <http://www.mpeg.org>
2. <http://www.xvid.org>
3. G.J.Sullivan, P.Topiwala, A.Luthra, "The H.264/AVC Advanced Video Coding standard: overview and introduction to the Fidelity Range Extensions", *SPIE conference on Applications of Digital Image Processing XXVII*, August 2004.
4. V. A. Chouliaras, J. L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso 'A multi-standard video coding accelerator based on a vector architecture', *Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2005)*, Las Vegas, Nevada, USA
5. D. Burger, T. Austin, 'Evaluating Future Microprocessors: The SimpleScalar Tool Set', <http://www.simplescalar.com>

Paper PC6: Ashwin K. Kumaraswamy, V. A. Chouliaras, T. R. Jacobs, and J. L. Nunez-Yanez, '*System-on-Chip Design Framework (SDF) unifying Specification Capture and Design Modelling*', Proceedings of the 2005 Electronic Design Processes (EDP) Workshop, April 6-8, Monterey Beach Hotel, Monterey, California, USA

System-on-Chip Design Framework (SDF) unifying Specification Capture and Design Modeling

Ashwin K. Kumaraswamy, V. A. Chouliaras, T. R. Jacobs, and J. L. Nunez-Yanez

Abstract— We propose a new EDA tool flow which aims to allow SoC architects to utilize an object-oriented approach in the development SoC's including shared-memory, cache-coherent, single-chip multiprocessors. The tool will allow the visual definition of a complex computation kernel/SoC through instantiation of parametric IP such as processors, SDRAM controllers, DMA engines, on-chip buses, switch matrices and coherency directories, coprocessors, etc. Such IP is captured either at the specification level via UML, at the model level (SystemC, SpecC or ANSI C) or at the implementation level (RTL VHDL or Verilog). The unified environment then simulates the whole system and in the process, a near-optimal solution in terms of area, power and performance, is achieved. Finally, the output of the tool consists of a cycle-accurate executable model accompanied by the system RTL.

from specification to SLDL(SpecC) and simultaneously work is being performed to accommodate SystemC, as its been widely used in the industry.

We have the SDF simulator which has been developed and this clearly is the basement for the existing mechanizations. The below figure is the overall flow of SDF

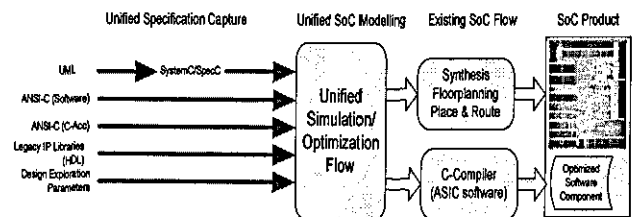


Fig 1. Overall SDF Flow

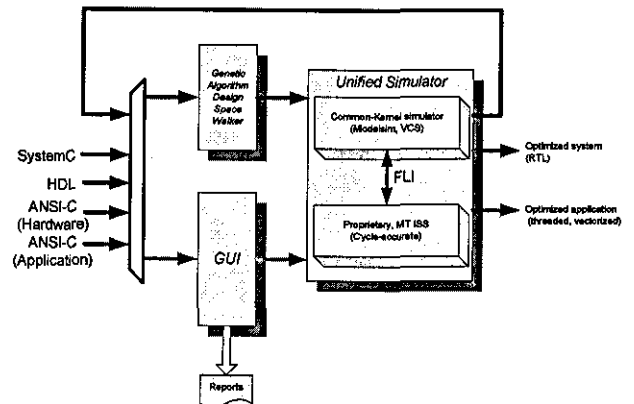


Fig 2: SDF Flow internal flow path

I. INTRODUCTION

We present preliminary results of a new EDA flow named System-on-chip Design Framework (SDF) which unifies the specification capture and design modeling. Current, tools in the market namely Incyte, Mageillem, Visual Elite have provided solutions for specification optimization, graphical design entry and hardware-software partitioning to help designing of high performance IPs, but clearly we are still lacking a complete robust flow which helps the designers to take designs from specification to silicon and there is been a concrete effort to develop such a flow.

SDF flow seeks to unify the specification capture, modeling, optimization of very high performing streaming system-on-chip designs through a unique combination of technologies. SDF is intended to be the future front end tool flow. The overall SDF flow can be classified into two parts namely the unified specification capture and the heart of SDF, the unified simulator.

At present the flow is partially completed with work being done on the specification capture stage so as to accommodate the existing system level design languages (SLDL) like systemc and specc.

We use UML as the front end specification capture format and convert the UML to a known SLDLs like SystemC and SpecC, this translation is being performed using a unique combination of technologies and we have a working model of translation kit

Affiliations

Ashwin K. Kumaraswamy, V. A. Chouliaras and Tom R. Jacobs are with Dept of Electronic and Electrical Engineering, University of Loughborough, Loughborough, UK, email: ashwin.k.ctes2004b@cse.london.com. J. L. Nunez-Yanez is with the Dept of Electronic Engineering, University of Bristol, Bristol, UK

II. UNIFIED SPECIFICATION

We propose a methodology that can transform UML models into a known System Level Design Language (SLDL) (SystemC/SpecC). In other words, UML model acts as a "wrapper" to the SLDL's methodology. In UML each aspect of the SLDL's methodology can be modelled and refined. This has various advantages. The standardization of UML provides a base to revise the approaches to combine SLDDL with object oriented analysis and design techniques (OOAD) techniques. One of the main directions for the joint application of SLDDL and UML can be identified as modelling SLDDL specifications with UML. This direction serves mainly the idea to make large SLDDL specification better understandable and to give additional information (e.g. inheritance hierarchies, dependencies, pattern structures) for documentation purposes or as additional implementation advice. UML is mapped onto the SpecC methodology. Uniqueness, to this new methodology, is that the UML representation of the system is

separated from the underlying methodology. This helps in unifying the ways a system can be represented in UML without worrying about the way it will be implemented. The reason behind using this approach is that the UML model can be ported seamlessly to any methodology. Thus we have to first understand how a system can be modelled in UML. Although there can be numerous ways of describing a system in UML, only one of these methods can be chosen. This way the code-generation (transformation) phase will be made easy.

A Hardware/Software co-designed system can be specified through the concepts of behaviours that interact via channels through ports and interfaces. There is a clear separation between computation and communication where behaviours model computation, and communication is modelled by using shared variables and/or channels [5]. Keeping this in mind, the first step is to decide on the modelling of the different aspects of a system, i.e. computation and communication. Computation will consist of behaviours and their definitions. Communication will consist of ports, channels and interfaces. (Interfaces can also be used in the modelling of computation [3, 8].)

Modeling of Computation

In UML, the behaviours are modelled as classes. The local variables and the functions are also modeled within the class in their respective positions. A composite behaviour will contain instances of other behaviours. These compositions can be modeled using associativity. When breaking down behaviour into sub-behaviours, for structural hierarchy, generalizations can be used. There can be two types of hierarchy: structural and behavioral. Structurally, behaviours can be broken down into sub-behaviours and these into sub-behaviours, and so on. Designs are specified in a hierarchical manner using top-down functional decomposition (behavioral hierarchy). Both these hierarchies correspond to the concept of generalization and associativity in UML [5, 8].

Modeling of Communication

To model interfaces, UML's interface notation is used. An interface is like an abstract class that consists of a set of *method declarations*. Interfaces can also be placed in a hierarchical fashion. Behaviours can, optionally, "realize" single or multiple interfaces. The channel or the behaviour that realize the interfaces should supply the definitions for the method declarations.

The stereotype, <<channel>>, is used to represent a class as a channel. Channels are also modelled in the same manner as behaviour. Ports can be modelled in two ways. A port can either be a simple variable or another Interface or Class. In order to identify an object as a port, the <<port>> stereotype is used. If the port is declared as a simple variable of type *type1*, the variable declaration in UML will be as

```
name: type1 <<port>>
```

The <<port>> stereotype helps in identifying certain variables and also associations as ports, rather than local variables or instances respectively.

Modeling of Execution

The main problem in designing a system is the modeling of execution or show parallelism i.e., to represent behaviors that will be executing in sequence, parallel or pipelined. There are two different ways of showing this. It is well known that in UML different views are meant for different activities of modeling. Thus, these considerations have to be mentioned in more than one of the views. In the static view (class diagram) we annotate these using stereotypes. This is very helpful, because the class diagram shows the static structure of the system. The problem of showing parallelism in the execution model can be solved through composition. Leaf behaviour, by itself will only perform its operations sequentially. If a component has to be modeled to execute in parallel or pipelined mode, then its behaviour can be further reduced into separate classes and its objects will be composed into the main component. These sub-behaviours can then be modeled to run in parallel or pipelined mode by specifying the mode of execution to the composite behaviour (main component). This can be done in the static view of the model. The actual execution of the composite behaviour can be modeled in detail, using State chart diagrams and/or Sequence diagrams.

It was concluded that the State Machine view and the Activity view of the UML had enough notations specified to describe the internal behaviours of any component. Clocks can also be modeled as behaviours and can be made to generate events. These events can be used in other views to specify the timing characteristics of the system.

Transformation of Static View

Since SpecC is not an Object oriented language, there is no way of representing object hierarchies. Thus generalization is used to model behavioral hierarchy. In other words, behavioral hierarchy is modeled as a composition of multiple behaviours, according to the SpecC methodology. Therefore generalizations are transformed in the same manner as associations. A static view is shown in figure (3).

Transformation of State Machine View

The state machine view describes the dynamic behaviour of objects. Each object is treated as an isolated entity that communicates with the environment by detecting events and responding to them [7]. A state machine is a graph of states and transitions. Usually a state machine is attached to a class and describes the response of an instance of the class to events that it receives.

A State Machine view is used to model the internal behaviour of an object of a class. A state machine contains states that are connected by transitions. Each state is defined as some unit of time in which the object stays and performs certain operations, whereas transitions are instantaneous, i.e. they occur at zero time. When an event occurs, it may cause the firing of a transition that takes the object to a new state. When a transition fires, an action attached to the transition may be executed. Theoretically, this execution period is zero. State machines are shown as a state chart diagram (Figure 4).

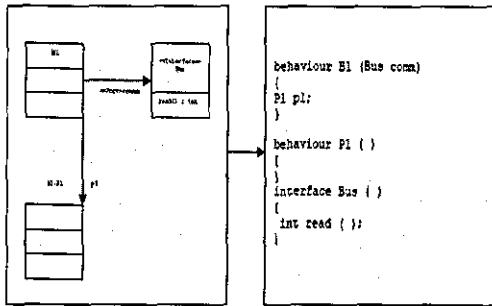


Fig 3: Static View

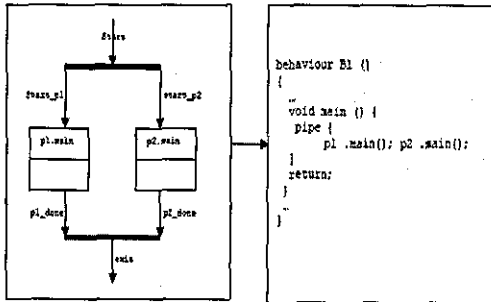


Fig 4: State Machine View

III. THE SDF FLOW

Fig.1,2 depicts a high-level view of the proposed SoC Design Framework tool that we are currently developing. It consists of the input interface which can accept silicon and software IP in a number of forms including SystemC, standard HDL (VHDL and Verilog), cycle-accurate C conforming to the SDF API and finally, standard C for the application. These elements are slotted in system-defined and user-defined 'Stencils' from which on they are available for manual or automatic instantiation and design space exploration.

A. Core Simulation Engines

The primary simulation engine is based around a parameterized, multi-context, Instruction Set Simulator (MT-ISS) derived from the Simple scalar computer architecture research tools [SS]. The default ISS has been re-architected to allow the instantiation of a number of processor contexts and additional programmer-visible state for multi-processor (MP) synchronization. The simulator can be considered as an Exclusive-Read, Exclusive-Write (EREW) Parallel RAM (PRAM) machine. Architectural hooks are in place to allow interfacing to a cycle-accurate (CA) back-end. In this way, the ISS is dynamically producing (short) instruction traces which are (dynamically) consumed by the CA back-end. In the process, various parameters are evaluated such as the Clocks-Per-Instruction (CPI) ratio per CPU, bus utilization, ICache and Dcache misses, pipeline stalls due to dependencies amongst others.

The MT-ISS is one of the core simulator of the SDF flow and drives both the programmable and non-programmable C-based simulation models along with being used for software development.

The second simulation engine is an industry-standard tool such as Mentor Graphics Modelsim. It interfaces to the cycle-accurate infrastructure via the FLI and allows for the modeling of legacy IP (VHDL, Verilog) and the primary output of the specification-capture front-end which is described in System-C

B. Manual and Automatic Flows

There are two major flow (feedback loops) in the SDF tool. The first is based around a GUI solution which is used to instantiate silicon IP blocks and application software components from the *IP stencils* on to the SoC *canvas*. The contents of those stencils can be 'dragged' onto the SoC area thus, incrementally building up and simulating the SoC model. We make no distinction as to whether the stencils contain synthesizable Silicon IP or CA models as the core simulators permit their arbitrary mix. This is of paramount importance in the modeling of highly-complex, future SoC architectures. Experimentation takes place after the SoC has been 'drawn' and its memory map established and populated. The feedback loop of Fig. 2 illustrates the manual or automatic refinement process, from SoC specification to performance closure and clearly illustrates the synergy between the MT-ISS, CA back-end and industrial simulators in providing a unified framework for SoC modeling.

A further route exists where the process is fully automated. In this case, a genetic-algorithm (GA) design space walker takes over the refinement process of the SoC once the initial allocation of programmable and non-programmable resources has happened.

C. Embedded CPU Stencils

The primary programmable engine used is based on an open-source, 32-bit RISC CPU with an extended Instruction Set to allow for hardware barrier synchronization. In addition, the programmer's model was extended to include a unique, non-programmable, processor ID field which is used to identify the executing CPU to a software thread.

As we are targeting primarily Data-Level-Parallelism (DLP), we have augmented the microarchitecture of the Leon-2 CPU to include a custom coprocessor channel in order to communicate to very high performance, tightly-coupled vector coprocessors [IEE Elec Letters], [IEEE ICCE]. Typical transactions along this new interface are depicted in the diagram of figure 5 shows a coprocessor data operation on cycle 1 followed by a host-to-coprocessor register transfer on cycle 2. In cycle 3, a coprocessor register is requested by the RISC processor but due to internal stall conditions, data are made available one cycle later than the expected time (cycle 5 instead of cycle 4). During that time, the main processor is held with the holdn signal. Finally, a second read operation, this time directed to Coprocessor 1, is initiated in cycle 6. Results are made available to the main pipeline in cycle 7.

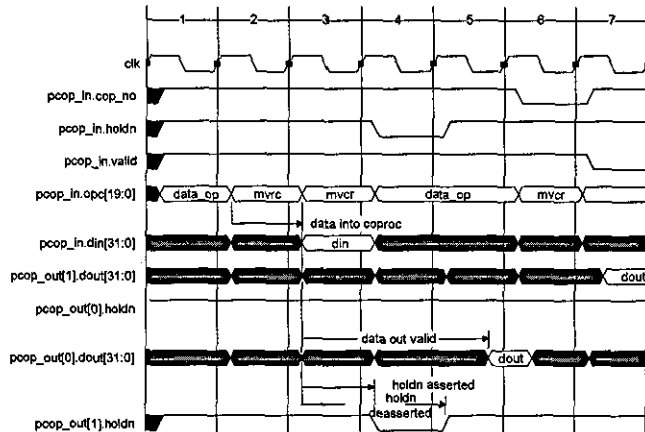


Figure 5: Typical Coprocessor Channel Transactions

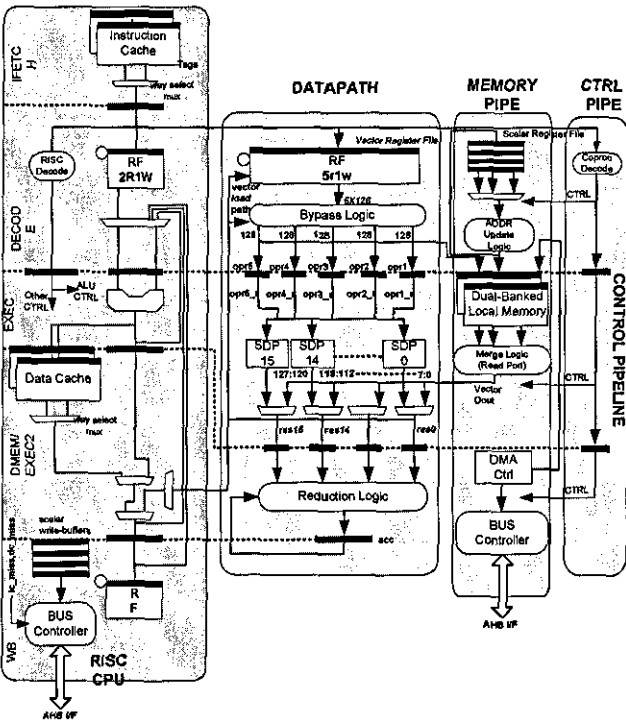


Figure 6: Scalar CPU and Vector Accelerator

Fig.6 shows the combined processor-coprocessor microarchitecture which includes a parametric vector accelerator implementing three custom instructions for the data-parallel sections of the MPEG-2 encoder attached to the scalar CPU which is a standard 5-stage design. From the diagram, instructions are fetched from the multi-way, set-associative instruction cache and clocked into the instruction register. Decoding takes place in the DECODE stage with the RISC register file accessed at the falling edge of the clock. The bypassing logic in DECODE determines whether register file data or internally pipelined results are clocked in the ALU input registers. During EXEC, the ALU operation is performed and a virtual address is computed. Scalar data cache access takes place during DMEM/EXEC2 and scalar results return to the RISC pipeline during this cycle. Finally, results are clocked into an intermediate register prior to committing to the

processor register file. The processor incorporates configurable data and instruction caches the former in a write-through configuration with no-write-allocate policy. Both caches are refilled over the on-chip bus via the bus controller.

D. Interconnect Stencils

We are targeting primarily the SoC modeling and implementation domain. We therefore have included support for multi-layer AMBA (AHB) [ARM] based on the infrastructure provided by the Opensource CPU, augmented with the hardware synchronization primitives. A typical scenario of a parametric, cache-coherent, SoC MP is depicted in Fig. 6.

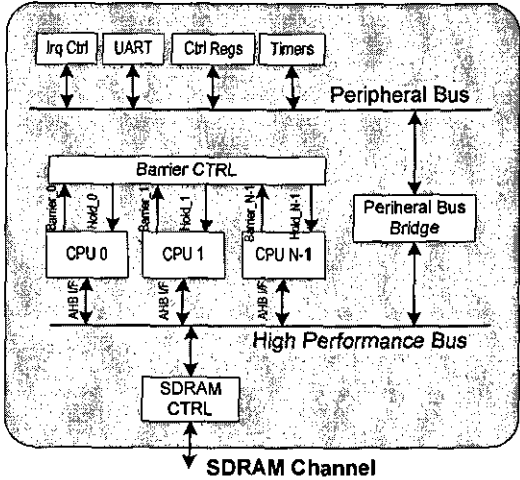


Figure7: Scalar, 32-bit RISC CPU pipeline

Further development is underway to allow for very high bandwidth interconnects (other than a hierarchy of buses) to be utilized. In this case, distributed coherency directors are utilized to ensure that the CPU caches remain consistent.

E. Streaming unit stencils

Prior research into statically-configurable (off-line configurable) vector/SIMD accelerators has successfully concluded that such units are of paramount importance in achieving performance closure in a consumer/media SoC. The complexity-metric reduction due to the vector instructions implemented via this tightly-coupled coprocessor is shown in a standalone identified

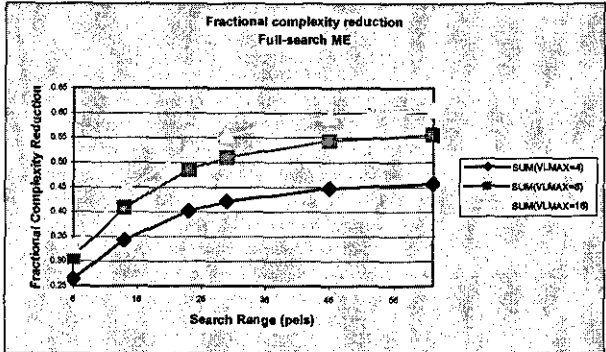


Figure8: MPEG-2 DLP benefit

IV. RESULTS

To establish the proposed methodology, we studied a multi-threaded implementation of the MPEG-2, TM5 reference video standard [mpeg.org]. The encoder was initially profiled in order to identify the most compute-intensive parts at function-level granularity. The complexity metric used was the dynamic instruction count of the application when compiled for a MIPS II-like CPU and executing on a single-context simulator.

From Fig. 7, the most compute-intensive function was identified as the inner loop of ME (*DIST1*). This function computes the error of the current macro block over all macroblocks in the search window of the reference frame and its complexity ranges from 52% to 73% of total dynamic instruction count for a search window of 7 to 63 pels respectively. The second most complex function was the forward-DCT computation (*FDCT*) with a complexity metric ranging between 2.1% and 21% of the total dynamic instruction count. *FullSearch* is the wrapper function around the low-level *DIST1* and implements the default ME algorithm. Its complexity ranged from 3.5% to 23.2% of the total complexity. This is the level at which we applied our threading technique as this allows the utilization of less complex, algorithmic ME methods such as three-step search [10] and four-step-search [11] in the parallelized encoder.

The performance of the threaded MPEG-2 encoder was evaluated in a relatively slow, vertical-moving sequence (Snowfall) and a very fast, circular-moving sequence (Rotating City). We used Full-Search ME which is the default algorithm in the reference MPEG-2 code.

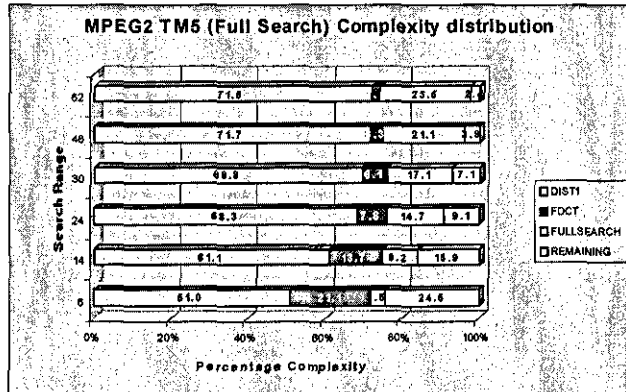


Figure 9: MPEG-2 TM5 Algorithm Profiling

Results depict the dynamic instruction count reduction for the primary processor context (thread 0) for the vertical and circular-motion video sequences respectively. Context 0 is the controlling thread in the parallel encoder as it performs I/O and activates the remaining threads early during execution and thus, suffers the maximum overhead.

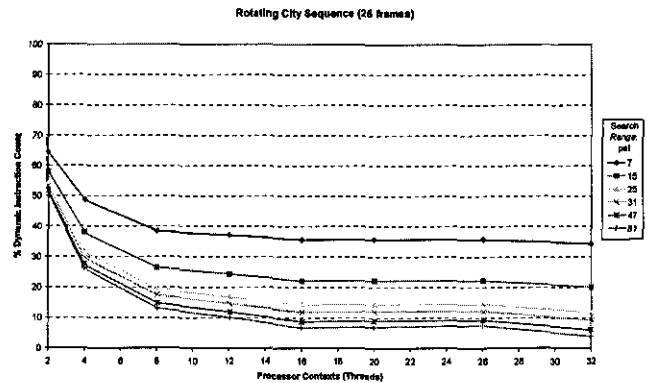


Figure 10: Theoretical Performance (Circular Motion)

Both graphs demonstrate a significant reduction in the complexity metric both when the number of processor contexts is increased and when the search window range is increased. The complexity improves no further once the number of processor contexts exceeds 32. Performance saturation at 95% complexity-metric reduction occurs when a threaded loop is executed only once. Results compare favorably with prior studies which achieved a 65%-70% complexity-metric reduction at a search range of 62 pels, for a 128-bit DLP architecture

V. COMPARISON OF LIKE MINDED EDA TOOLS

Tools	What it does	What it does not
Incyte	1.Specification Optimization System 2.Designer specification oriented 3.Tool shows the potential problems in the design thus helping in fast design time	1.Design Modelling in not done 2.No Hardware-software partitioning 3.Not really a integrator platform based on any Bus standard
Magilleium	1.Graphical Design Entry based tool 2.Integration Platform based on AMBA 3.Transaction RTL Builder, full support for SystemC 4.Good Verification system in place	1.No option for design modelling and design exploration at the highest level. 2.Basically choosing blocks from the existing library
Visual Elite	1.Graphical Design Entry based 2.Performs Hardware-Software partitioning 3.Helps in design using specific Microprocessors	1.Not a design specification capture and design modelling based tool
System-on-chip Design Framework	1.Graphical Design Entry 2.Design Specification capture and architecture exploration along with direct choice from legacy IP library. 3.Integration platform with AMBA 4.Hardware-Software partitioning. 5.Supports SystemC,C RTL, SpecC	

VI. CONCLUSION

We have demonstrated the new EDA flow proposal and submitted preliminary results of the SDF flow that we have developed on two fronts.

1. UML diagrams being converted to the desirable SLDL.
2. The simulation performance of the SDF unified simulation flow.

VII. FUTURE WORK

To integrate the tool flow so as to make the automation, we have been able to demonstrate that such a tool flow is conceivable.

- Future work will quantify on cycle effects of the bus-based configuration as well as the benefit of local scratchpad memories over the parametric Data Cache.
- Complete the UML to SystemC in the same lines of SpecC
- Develop the GUI for graphical design entry.
- Develop the automation as envisioned.

REFERENCES

1. M. Keating and P. Bricaud, "Reuse Methodology Manual for System-on-Chip designs, 2nd Edition, Kluwer Academic Publishers, Norwell 1999.
2. F. Balarin *et al.*, "Hardware-Software Co-Design of Embedded systems, The POLIS approach," Kluwer Academic Publishers, 1997.
3. D. Gajski, J. Zhu *et al.*, "SpecC: Specification Language and Design Methodology", *Kluwer Academic Publishers*, 2000.
4. Rainer Dömer, Daniel D. Gajski, Andreas Gerstlauer, "SpecC Methodology for High-Level Modeling," 9th EDP IEEE/DATC Electronic Design Processes Workshop 2002.
5. Object Management Group, Omg unified modeling language specification version 1.3, June 1999.
6. D. E. Lackey, "Applying Placement-based Synthesis for On-time System-on-a-Chip Design", *IEEE Custom Integrated Circuits Conference*, 2000, pp. 121-124.
7. Object Management Group, Omg-xml metadata interchange version 1.2, January 2002.
8. J. L. Diaz-Herrera, An isomorphic mapping for SpecC in UML, Internet: <http://ist.unibwmuennen.de/GROOM/OMER-2/papers/OMER2->
9. DiazHerrera.pdf, 2000, SPSU-CS TR 2000.
10. Sikora T, "MPEG Digital Video--Coding Standards," *IEEE Signal Processing Magazine*, Vol. 14, No. 5, September 1997, pp. 82-100.
11. Motion Picture Experts Group <http://www.mpeg.org>
12. V. A. Chouliaras, J. L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso 'A multi-standard video coding accelerator based on a vector architecture', Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2005), Las Vegas, Nevada, USA
13. Shen K, Delp E J, "A parallel implementation of an MPEG encoder: faster than real-time!", In Proceedings of the SPIE Conference on Digital Video Compression: Algorithms and Technologies, pp. 407-418, San Jose, California, 5-10 February. 1995.
14. "The Leon-2 processor User's manual, XST edition, ver. 1.0.14", <http://www.gaisler.com>
15. Theo Ungerer, Borut Robič, Jurij Šilc, "A survey of processorw with explicit multithreading", *ACM Computing Surveys (CSUR)*, Volume 35 Issue 1, March 2003
16. SimpleScalar LLC <http://www.simplescalar.com/>
17. Martinez J. F., Torrellas J "Speculative synchronization: applying thread-level speculation to explicitly parallel application" *ACM SIGARCH Computer Architecture News*, 30, 5, pp.18-29
18. Zeng and Liu "A new 3 step search Algorithm for Block Motion Estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 4, No 4, Aug. 1994.
19. Lai-man Po and Wing-Chung Ma, "A novel four step-search algorithm for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 313-317, 1996.

Paper PC7: V M Dwyer, S Agha and V. Chouliaras, '*Low Power Full-Search Block Matching using reduced bit SAD values for early termination*', Proceedings of Mirage 2005 International conference on Computer Vision/Computer Graphics collaboration techniques

LOW POWER FULL SEARCH BLOCK MATCHING USING REDUCED BIT SAD VALUES FOR EARLY TERMINATION

V M Dwyer, S Agha and V Chouliaras

v.m.dwyer@lboro.ac.uk

Loughborough University, Dept. of Electronic & Electrical Engineering, Ashby Road,
Loughborough, Leicestershire, UK

ABSTRACT

Full-search motion estimation is often employed for selection of the best motion vector through a minimum SAD by iterating over all candidate motion vectors of the search area. However, although the dataflow is regular and the architectures straightforward, the computational complexity is high. Considering all possible candidate motion vectors and calculating a distortion measure at every search position produces a high computational burden, typically 60-80% of a video encoder's computational load. This makes it unsuitable for real time video applications. To alleviate the problem SAD calculations based on reduced numbers of bits (RBSAD) have been suggested which gives power and time savings, but the reduced dynamic range means that picture quality can be compromised. This current work presents a corrected-RBSAD algorithm, which corrects to full SAD resolution under certain circumstances. The compromise achieved provides the low power of the reduced bits and a higher accuracy, closer to that of a Full Search.

1. INTRODUCTION

Battery powered real-time visual communication applications place stringent requirements on power consumption. As a result Motion Estimation algorithms [1, 2], which eliminate the temporal redundancy in video sequences, are widely used in video coding. Currently the prevailing method of Motion Estimation has been the block-matching algorithm [3], which computes a motion vector on a block-by-block basis, as it generally outperforms other methods such as the pel-recursive algorithm [4]. The block-matching algorithm divides the current frame F_c into non-overlapping square blocks of $N \times N$ pixels which are matched to blocks of the same size in some region of a reference frame F_r . This region is known as the Search Area and is predetermined by the search strategy adopted. All pixels within the same block of the current frame are assumed to have the same motion vector. N is most commonly taken equal to 16, and

the Search Area is generally defined as a square of size $(N+2p)^2$ surrounding the position of the block of interest referenced to the frame F_r . With billions of arithmetic operations per second [5, 6] and a memory bandwidth of the order of GByte/s, it is generally not feasible to search all possible positions in the Search Area for each block in each frame (the Full-Search Motion Estimation algorithm [3]), and encode a CIF sized video at 30 fps, with the low power constraints of today's processor technology, particularly if the search range (effectively the value of p) is large [7].

Fast motion estimation algorithms can give reduced computational complexity, as well as advantages for VLSI design in terms of area and power consumption [e.g. 5 and references therein, 6]. However, the reduced computational complexity gained by these fast Motion Estimation algorithms has often to be offset by losses in visual quality and/or by irregularities in data flow. Consequently it is difficult to achieve efficient VLSI implementations that can employ Data Reuse efficiently [5,6].

The figure of merit used to determine the 'best match' between blocks in the current and reference frames is usually a distance metric, typically the Sum of Absolute Difference (SAD) between the current frame Micro-Block (MB) and a candidates MB in the reference frame. This is generally termed the SAD value which, taking the reference frame as the immediately preceding frame, may be written as

$$SAD(m,n) = \sum_{i,j=1}^{16} |s(i,j,k) - s(i+m,j+n,k-1)| \quad (1)$$

for a 16×16 MB. The best motion vector is determined as

$$[u,v]^T = \arg \min_{m,n} SAD(m,n) \quad (2)$$

Here $s(i,j,k)$ is the (8-bit luminance) pixel value at (i,j) in frame k , u and v are the horizontal and vertical motion vectors respectively, and the minimization is performed

over the set of candidate blocks in the Search Area. For the Full Search method, SAD values are computed for all blocks in the designated region [3] and the search naturally is computationally burdensome. Even with p as small as 8 this still corresponds to 256 possible candidate blocks. One means of reducing the computation is to use one of the fast motion estimation methods such as the Three Step Search and its variants [8, 9], the conjugate gradient method [10], and many others which may be found, for example, in refs [5, 6] and references therein. Such 'fast' methods base the candidate set on previous results which increases the design complexity and disturbs the dataflow making dedicated hardware implementations very difficult, however power and speed savings are envisaged [5, 6].

The usual means of testing the accuracy of these faster algorithms is through the use of the Peak-Signal to Noise Ratio (PSNR) which is usually defined as the fractional RMS error between the predicted and true frames expressed on a dB scale. To be considered realistic, any fast algorithm really needs to be within a fraction of a dB of the Full Search (the gold standard) value.

In order to preserve the dataflow regularity in its VLSI architecture, and at the same time to reduce computational complexity, a number of authors have employed a full-search algorithm, but with eqn (1) based on a reduced number of bits. Such schemes are known as Bit Truncation and provide an RBSAD (Reduced-Bit Sum of Absolute Difference) metric or distortion measure [11, 12]. The downside of such a method is the reduced dynamic range. In the case of a 4-bit RBSAD the resolution for pixel luminance values in the integer range [0, 255] is reduced from unity to steps of 16. The result is that although two blocks may provide the best match with the reduced resolution, it is possible that the match with full resolution may not be best. This leads to an increased error matrix and consequently a lower bit-rate, or a higher quantisation error and, possibly, a reduction of visual quality. Whilst these problems are possibilities, in practice the PSNR for the two methods for real sequences are very close, Table I.

The only case studied in which the average PSNR for the sequences studied is significantly worse using RBSAD is the "Claire" sequence where the average PSNR values (with full and reduced resolution) are already very large. A typical frame from the sequence is shown in Fig.(1). The purpose of this work is to investigate means of providing a correction to only those cases in which the RBSAD is poor, so that the advantages of reduced bits (power and speed) can be maintained without some of the disadvantages. This means that correction should largely be restricted to the "Claire" se-

quence. The difference between the average PSNR for the full resolution and with reduced bits (using 4-bits) in the case of the "Claire" sequence is around 2dB. This difference is plotted for the first one-hundred frames in the sequence in Fig.(2). The problem is that with this "head and shoulders" sequence there are several very good matches for each current frame MB and the RBSAD method has difficulty in selecting the best. Indeed the RBSAD calculation will not be able to distinguish between any candidate blocks for which the RBSAD calculation is equal to zero, and this still leaves a wide range of possible SAD values. An obvious and simple correction to the, generally very good, RBSAD method is to revert to a full resolution value whenever $RBSAD = 0$.

Seq.	Mom	Fore-Man	Fog	Snow Fall	Snow Lane	Claire
FSSAD	35.86	28.46	32.28	26.82	30.16	46.15
RBSAD	35.58	28.37	31.76	26.70	30.09	43.29
Corrected RB	35.58	28.37	31.76	26.70	30.09	44.09
% recalculation	1.75	3.2	1.79	0	6.65	23.41

Table I PSNR for full resolution (FS) and reduced-bit (RBSAD) algorithms.

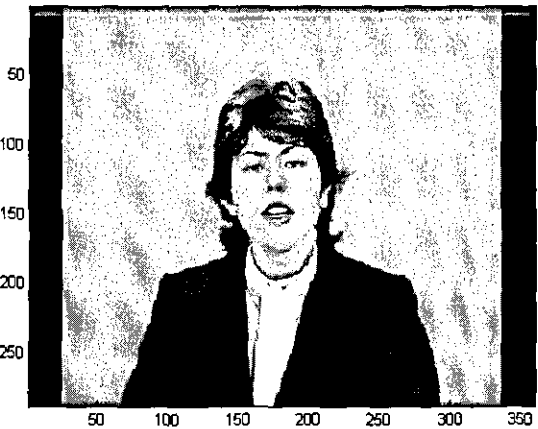


Figure 1. Frame 11 from "Claire" sequence

The hardware implications of such a scheme are obvious as whenever $RBSAD = 0$, the full resolution calculation is simply the 4-bit RBSAD of the lower bits. Consequently the calculation may be split into two. A 4-bit

RBSAD calculation for the upper four bits, which is generally used, and, in the case that this upper bit $RBSAD = 0$, a 4-bit RBSAD calculation for the lower four bits. For the hardware, two options are possible. Either: (i) The two 4-bit calculations are pipelined and only in the case when $RBSAD = 0$ is the correction to full resolution applied; or (ii) Two identical Full Search hardware layouts are created each based on 4 bits. The first runs continuously and the second performs a correction whenever the RBSAD value calculated by the first is zero. The result, in either case, is a saving in power and speed.

A second, but not inconsiderable, power saving, which results from this architecture, is in memory reads from on-chip memory. If the data is stored in two separate memory blocks as it is written to the on-chip memory, with the upper four bits in one block and the lower four bits in another, then typically only reads from the upper block will be required. Only when the upper four bit $RBSAD = 0$, will require reads from both memory blocks will be necessary.

The purpose of this paper is to investigate the accuracy of this simple correction to the standard reduced bit method and to investigate whether such correction methods in general are appropriate.

2. ALGORITHM

We consider here a number of test sequences, as shown in Table I. These involve most types of motion seen in typical sequences and so are representative of what might be expected in real video clips. We imagine a mechanism of thresholding, in which an RBSAD calculation is converted into a full resolution SAD if the value

of RBSAD is less than some value T . In this case the algorithm we shall present actually corresponds to the case $T = 1$. Thus we effectively execute the pseudocode

```
If (RBSAD(7:4)) < T then
    output = SAD(7:0)
else
    output=RBSAD(7:4)
end;
```

and suppose that, in some way as yet unspecified, the correction from reduced bit to full resolution can be achieved in hardware.

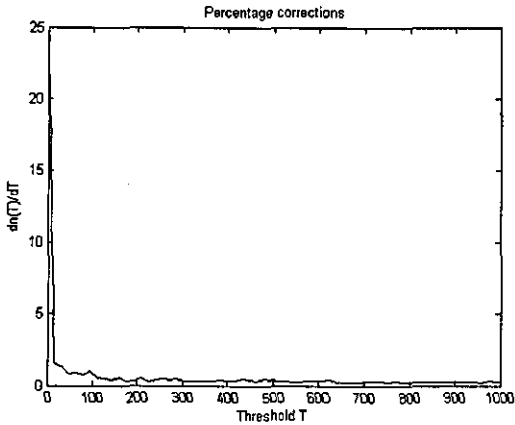


Figure 3. The fraction of corrections $dn(T)/dT$ made with a threshold value of T for the "Claire" sequence.

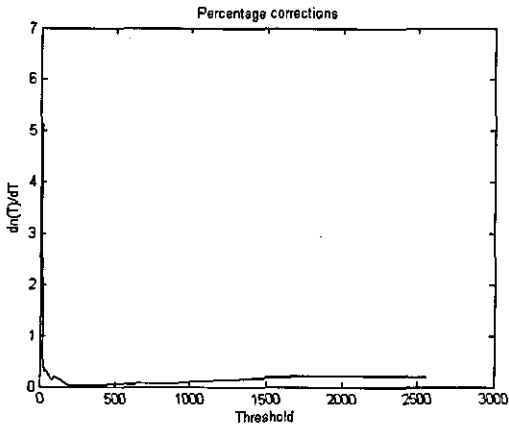


Figure 4. As Fig. (3) for the "snow lane" sequence.

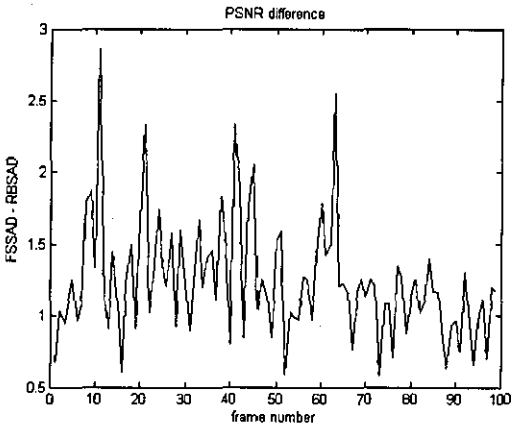


Figure 2. Difference between full resolution (FSSAD) PSNR and reduced bit RBSAD PSNR values.

Note that here we have used the notation that $RBSAD(7:4)$ is an SAD calculation based on using the top four bits (7:4) of the luminance pixel values.

In a motion estimation calculation for a given sequence, the fraction of times $n(T)$ that the RBSAD calculation fell below T was recorded as a function of T , this gives a measure of the efficiency of the method for that threshold value. Fig. 3 & 4 shows the derivative dn/dT for the "Claire" sequence and the "snow lane" sequence. It is clear that dn/dT is large at $T=1$ rapidly saturating for larger threshold values. For the "Claire" sequence dn/dT shows a rapid increase as $T \rightarrow 0$ and similar analysis of other sequences shows the same behavior although to a lesser extent. This implies that if we apply a correction to the RBSAD to get the full resolution SAD for $T = 1$, i.e. whenever we get an RBSAD value = 0, we shall correct for many of the errors in an RBSAD calculation.

The upper four bit RBSAD = 0 occurs only when

$$|s(i, j, k) - s(i + m, j + n, k + 1)| < 16$$

for all i and j in the current frame MB. This means that the restriction of the analysis to only the upper four bits (7:4) evaluates to zero. I.e.

$$|s(i, j, k)_{7:4} - s(i + m, j + n, k + 1)_{7:4}| = 0$$

for all i and j in the current frame MB. As a consequence only the lower four bits contribute to the SAD sum and we may replace the full resolution SAD value by the lower four-bit calculation. Thus

$$\begin{aligned} &|s(i, j, k) - s(i + m, j + n, k + 1)| \\ &= |s(i, j, k)_{3:0} - s(i + m, j + n, k + 1)_{3:0}| \end{aligned}$$

for all i and j in the current frame MB.

This then corresponds to executing the pseudocode

```

If (RBSAD(7:4)) = 0 then
    output = RBSAD(3:0) = SAD(7:0)
else
    output = RBSAD(7:4)
end;
    
```

where the RBSAD(3:0) value is computed from an identical replica of the hardware (or indeed even the same hardware) to that which obtained the value of RBSAD(7:4). The calculation of the two stages can either be pipelined in the same unit or performed in parallel by separate units. Either way, depending on the number of corrections made, a considerable saving in computation and hence power may be achieved. Most of the current 8-bit (full resolution) architecture designs may be used as they are typically created in a bit slice fashion.

The zero flag of the final add-and-accumulate adder, at the base of the adder tree use to evaluate eqn. (1), is

used as an enable/disable for the lower bit calculation and also as a control for the MUX which selects between the upper and lower bit values, Fig. 5.

Assuming that a four bit RBSAD takes roughly half the power of a full resolution SAD calculation, if a recalculation is required 23.4% of the time (Table I), a power saving of roughly $76.6/2\% = 38.3\%$ is obtained compared to the Full Search method. Note that in the cases for which the RBSAD calculation produces an accurate PSNR value (i.e. all but the "Claire" sequence) there are almost no corrections applied. In addition, either with only a minimal increase in area the timing can be roughly halved or alternatively with similar timings the silicon area can be roughly halved.

3. ARCHITECTURE

The architecture envisaged involves a number of Processing Elements (PEs), each of which deals with the calculation of the best fit of a single MB in the current frame.

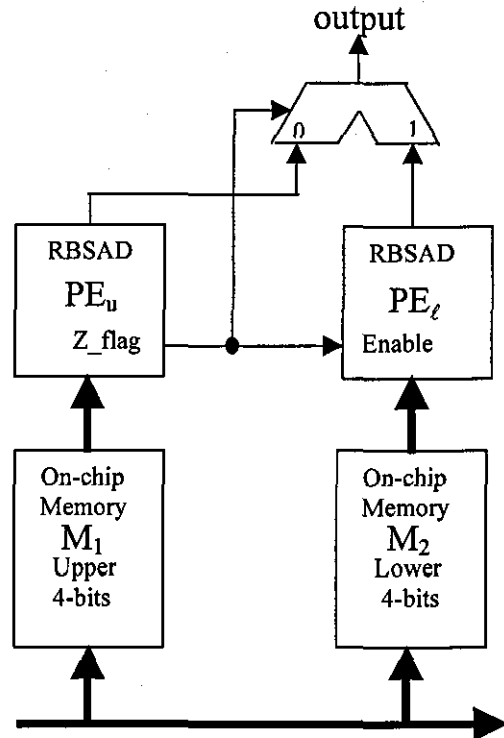


Figure 5. Upper(PE_u) and lower (PE_l) SAD calculations in a processing element PE.

Thus an image with dimensions 320×480 pixels will involve 20 PEs. PE₁ will work on the current frame data in columns 1 to 16, PE₂ will work on columns 17 to 32 etc. The RBSAD(7:4) calculation takes data from on-chip memory M1, which stores the upper four bits of the pixel values as they are read in from external memory. Unless RBSAD(7:4) = 0, the data in on-chip memory M2, which stores the lower four bits, will not be read.

Not having to read all the data from on-chip memory can yield a large saving in power requirement. In addition using only the top four-bits yields faster adders (for four bit inputs standard ripple adders are as fast as the so called 'fast adders' and consume much less area) and consumes significantly lower power, Fig. 5. A particular architecture has not been specified at this time as many Full Search architectures apply. The simple implementation suggested above requires that a single PE calculates the best-fit for a single MB, and the architecture would need significant modifications for those designs for which many PEs work in parallel comparing candidate MBs in the search window for a single MB in the current frame. Nevertheless such modifications are clearly possible.

4. CONCLUSIONS

In this paper we have proposed an alternative implementation to the standard Full Search Block Matching algorithm for the estimation of Motion Vectors in video sequencing. The method is based around the general good performance of the method of Bit Truncation in which typically the upper four bits are used in the Sum of Absolute Difference calculation. It has all the advantages of the reduced bit method in that the Reduced Bit SAD (RBSAD) may be calculated with power and time savings, but essentially uses the calculated RBSAD value as an early termination of the calculations. For the "Claire" sequence, where the RBSAD value is significantly lower than the SAD value, we have shown that the majority of fixes for the RBSAD value occur when the match is good and RBSAD(7:4) = 0. Correcting the SAD output to the true value SAD(7:0) is possible by repeating the calculation on the lower four bits, i.e. RBSAD(3:0). On the other hand for the other sequences, in which the values of SAD and RBSAD are very close, virtually no extra computations are required.

The majority of standard architectures based on a bit slice design and a single processing element per current frame MB can be easily implemented with this method. The saving in PSNR, shown in Fig. 6 for fifty frames of the "Claire" sequence and in Table I for a group of other

sequences, shows a significant improvement in accuracy towards the Full Search method but with most of the power and time savings of the Reduced Bit SAD method. Effectively the RBSAD is used as an early termination criterion.

The analysis presented here is based upon a correction being applied only when RBSAD = 0, i.e. $T = 1$. It is possible to extend the technique to correct the RBSAD to a full resolution value for other threshold values or, for example, in the case of small motion vectors. This will be considered in a future publication [13].

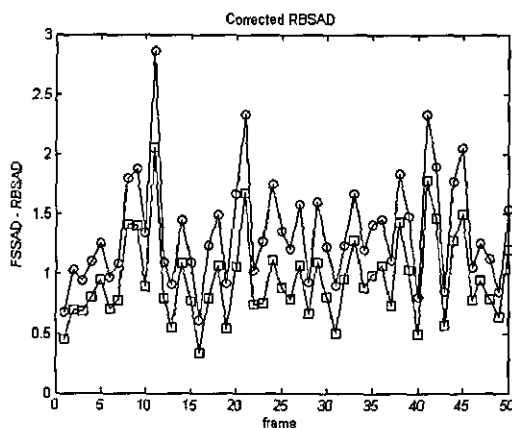


Figure 6. Difference between FSSAD and RBSAD. The circles represent the standard RBSAD while the squares represent the corrected curve.

5. REFERENCES

- [1] ISO/IEC JTC1/SC29/WG11 1313-1, "Coding of moving pictures and associated audio," 1994.
- [2] CCITT SG XV, "Recommendation H.261 - Video codec for audiovisual services", 1990.
- [3] J R Jain and A K Jain, "Displacement measurement and its application in interframe image coding," IEEE Trans Commun, COM-29 1799-1808 (1981).
- [4] A Netravali and J D Robbins, "Motion compensated television coding: Part I", Bell Syst. Tech J., 58 629-668 (1979).
- [5] P M Kuhn, "Fast MPEG-4 Motion Estimation: Processor based and flexible VLSI implementation" J. VLSI Signal Processing 23 67-92 (1999).
- [6] P M Kuhn, G Diebel, S Herman, A Keil, H Mooshofer, A Karp, R Mayer and W Stechele, "Complexity and PSNR-comparison of several fast Motion Estimation algorithms for MPEG-4", SPIE 3460, Applications of Digital Image Processing XXI, San Diego, USA, 486 - 499 (1998).
- [7] V G Moshnyaga, "A new computationally adaptive formulation of block-matching Motion Estimation", IEEE Trans. Circuits Syst. Video Technol. 11, 118-124 (2001).

- [8] T Koga, K Lumina, A Hirano, Y Lijima and T Ishiguro, "Motion compensated interframe coding for video conferencing", *Proc NTC 1981*, G5.3.1-5 (1981).
- [9] H Jong, L Chen and T Chieuh, "Accuracy improvement and cost reduction of three-step search block matching algorithm for video coding", *IEEE Trans. Circuits Syst. Video Technol.* **4**, 88-91 (1994).
- [10] R Srinivasan and K Rao, "Predictive coding based on efficient Motion Estimation", *IEEE Trans. Commun.*, **38** 950-953 (1990)
- [11] Y Baek, H S Oh and H K Lee, "An efficient block-matching criterion for motion estimation and its VLSI implementation", *IEEE Trans Consumer Electron.*, **42** 885-892 (1996).
- [12] S Lee, J-M Kim and S-I Chae, "New Motion Estimation algorithm using an adaptively quantized low bit-resolution image and its VLSI architecture for MPEG2 video encoding", *IEEE Trans. Circuits Syst. Video Technol.* **8**, 734-744 (1998).
- [13] V M Dwyer, S Agha and V Chouliaras, in preparation.

Paper PC8: Tom R. Jacobs, **Vassilios A. Chouliaras** and Jose L. Nunez, '*A Thread and Data-Parallel MPEG-4 Video Encoder for a System-On-Chip Multiprocessor*', accepted for oral presentation at the IEEE 16th International Conference on application-specific architectures and processors (ASAP 2005), Samos, Greece, July 23-25 2005

A Thread and Data-Parallel MPEG-4 Video Encoder for a System-On-Chip Multiprocessor

Tom R. Jacobs, Vassilios A. Chouliaras
Department of Electronic and Electrical Engineering
University of Loughborough, Loughborough, UK
t.r.jacobs@lboro.ac.uk

Jose L. Nunez-Yanez
Department of Electronic Engineering
University of Bristol, Bristol, UK
j.l.nunes-yanez@bristol.ac.uk

Abstract

We studied the dynamic instruction count reduction for a single-thread, vectorized and a multi-threaded, non-vectorized, MPEG-4 video encoder. Results indicate a maximum improvement of the order of 88% for 22 CPU contexts for the multi-threaded case whereas the single-thread, vectorized version demonstrates an 85% improvement for a vector register file length of 24 bytes, over the scalar case. We present VLSI macrocells of a vector accelerator implementing a subset of the MPEG-4 vector ISA and a 2-way, parametric, bus-based, cache coherent, SoC multi-processor.

1. Introduction

As the demand for video and multimedia products continues to expand, the problem of transmitting and processing the ever-increasing amount of media content becomes more acute. To address this, a number of lossy video compression standards such as MPEG-1 [1], MPEG-2 [2], MPEG-4 [3] and H264 [4] have been developed in recent years, with each method being more sophisticated and complex than earlier approaches. One method of providing the computational power for these increasing complex standards has been to create very powerful, small form-factor System on Chip (embedded) computer systems to ensure real-time video content delivery in every day consumer products. A very potent leverage of SoC computation power comes from the exploitation of various degrees of parallelism available in these standards the most prominent of which are Data-Level Parallelism (DLP) and Thread-Level Parallelism (TLP). These types of parallelism can be exploited individually by vector/SIMD architectures [5] and Chip-Multiprocessing/Multithreading respectively.

This work addresses the extraction of DLP and TLP and quantifies the performance benefits (dynamic instruction count reduction) in an open-source, integer-

only implementation of the MPEG-4 standard [6]. This paper will show the significant reduction in per-CPU instruction count achieved by exploiting both forms of parallelism. The paper is structured as follows: Section 2 describes the MPEG-4 XviD video compression standard, identifies the most computationally-expensive functions and discusses the potential of DLP and TLP. Section 3 details the abstract methodology we developed to parallelise arbitrary workloads and Section 4 applies this methodology to MPEG-4. This is followed by a discussion of a novel, multi-processor simulator developed for this purpose. Section 6 presents the results, in terms of dynamic instruction count reduction, for the non-vectorized, threaded and single-thread, vectorized versions of the workload. VLSI macrocells for the DLP-accelerator and a System-on-Chip (SoC) multi-processor are presented and finally, this work concludes by consolidating our findings and identifying issues to be studied in the future.

2. MPEG-4

The MPEG-4 standard was designed for low-bandwidth multimedia applications. The specification addresses not only video compression but also the use of audio, 3D objects and interactive modules. Video compression is an important aspect of the standard and this work focuses primarily at the hardware-software interface of MPEG-4 video coding and high-performance embedded computing platforms.

As in previous MPEG standards the specification standardizes the bitstream structure and not the strict implementation steps towards producing that bitstream. This extra degree of freedom allows for multiple implementations of the standard with one such implementation being the integer-only, open-source XviD.

In a logical progression from older MPEG standards such as MPEG-1 and MPEG-2, MPEG-4 is a discrete Cosine transform (DCT) and motion estimation (ME) based video compression scheme. After colour

conversion into YUV colour space each frame is divided into 8x8 blocks and the chrominance component sub-sampled. These blocks are grouped into macroblocks containing 4 luminance blocks and a corresponding number of chrominance blocks, depending on the colour scheme chosen (4 for 4.4.4., 2 for 4.2.2 and 1 for 4.2.0). DCT is performed on both the luminance and chrominance components for an intra-coded frame. This transforms the frame content from the spatial domain to the frequency domain in which higher frequency information, to which the human eye is less sensitive, is removed via more coarse quantisation. The net effect is the production of a compressed bitstream by eliminating spatial redundancy within the image.

For temporal (inter-frame) redundancy, motion vectors (MV) are computed by tracking areas of high similarity from one frame to the next. The XviD encoder searches exhaustively over a finite area within previously encoded frames in search for a macroblock that matches best the current macroblock. This matching process is via the *Sum-of-absolute-differences* (SAD) mechanism and proceeds until a macroblock, within a prescribed error limit, is found. The residue error produced from ME and the reconstructed frame in the encoder (produced after motion compensation, MC) is transformed and quantised in the same manner as for intra frames.

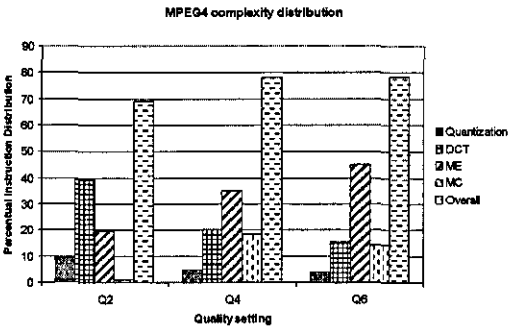


Figure 1: MPEG-4 (XviD) profiling

Figure 1 depicts the most computationally-expensive operations in the encoding process for a number of Quality levels. These quality levels enable various features of the encoder such as half pixel resolution MV and inter4v (inter encoded MB using four MV for MC, one per 8x8 luma block), with more features introduced as the quality level increases. The results of the profiling were obtained through measuring the number of instructions executed and mapping them to the four functional groups above. To allow for the real-time implementation of streaming-

video in small form factor consumer appliances it is clear that these functions should be primarily targeted for parallelism extraction at all levels.

3. TLP Methodology

Parallelising code at the thread level involves distributing the statically-threaded control-flow-graph (CFG) of the application across the active processor contexts (CPUs) within the shared-memory system. In the proposed approach, the most compute-intensive functions are manually selected after profiling and threaded at loop-level with different iterations of loops assigned to different CPUs while fully obeying the algorithm data-flow-graph (DFG). Figure 2 depicts diagrammatically this process.

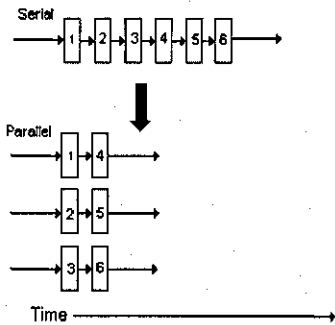


Figure 2 Transformation of loop from serial to parallel

From the figure, a 6-iteration loop is evenly split across three processor contexts. The process of converting from the single threaded model into the multi-processor one involves a number of distinct steps with an optional extra step depending on the number of available processors:

- Firstly all data dependencies between loop iterations need to be addressed. This is achieved by evaluating each functional block within the iterations, assessing its data requirements and if necessary segmenting the loop into serial and parallel sections.
- Once all data dependencies have been resolved the most evident step is to reduce the number of iterations of the loop is reduced and the reduction is calculated using equation 1:

$$parallel_iterations = \frac{serial_iteration}{processor_count} \quad [1]$$

- Since all processors are working in an iteration subset of the original loop, a method of mapping the non-threaded iteration range to the that of the threaded case is needed:

$$\text{original_iteration} = \text{processor_number} + \\ (\text{parallel_loop_iteration} \\ * \text{processor_count}) \quad [II]$$

- Due to multiple processors running simultaneously the scope of programming objects (variables) in each processor and their exclusivity is of great importance. To achieve exclusivity, processors work with private (local) variables. To access shared arrays, the later should be declared as global (which is very inefficient) or static, in the current scope.

So far it has been assumed that equation I produces an integer iteration number. This is not always the case and additional code is needed to support this situation. This extra code is known as the 'remainder' code and has a direct resemblance to the process of strip-mining in vectorized applications..

4. TLP extraction in MPEG-4

The loops associated with the transform functionality are located within the encoding functions for both I and P/B frames, *FrameCodeI* and *FrameCodeP* respectively. Both loops encompass transform (DCT), quantisation (Q) and VLC functionality. In the single-thread case the encoder scans each row of the frame, macroblock by macroblock, executing the same subset of code. In the multi-threaded environment the entire macroblock row is the shared component with all processor contexts encoding the row in parallel, one MB per CPU.

4.1. FrameCodeI

When encoding an Intra frame the pixel data is transformed and quantised along with its inverse within *MBTransQuantIntraMT*, a modified version of *MBTransQuantIntra*. Following that the quantised data is VLC-coded in function *MBCoding*. To allow *MBTransQuantIntra* to be executed in parallel, one extra variable, a per MB quant value, need to be passed into the function and additional code is needed to support this extra variable. This quant value was originally obtained from *pEnc->current->quant*, and describes the current quantisation level of the frame and changes with each additional macroblock encoded. Due to this dependency, each macroblock requires a

private quantisation value. This translates to a serial loop, within the parallel section, required to calculate a private quant value for each MB in the row before *MBTransQuantIntraMT* is called.

4.2. FrameCodeP

The process of encoding P and B frames is split into two parts in function *FrameCodeP*. First, motion estimation is carried out on the whole frame by calling function *MotionEstimation*. Within this function the motion vectors for each macroblock are calculated. With all the motion vectors calculated, the process of transforming and encoding each block then follows. As with the I frames case, rows of macroblock are encoded in parallel.

Another major task in the encoding process is MC which re-establishes macroblock pixel data from the motion vectors computed previously. This takes place in function *MBMotionCompensation* and can be executed in parallel since there are no data dependencies across macroblocks. Once the pixel data has been recreated a similar routine as that for I frames is executed with a serial section for calculating the frame quantisation parameter. These values are passed into the parallel-executing *MBTransQuantIntraMT* function.

ME, like transform, executes in parallel, per macroblock row. Within the *MotionEstimation* function the process of motion-vector (MV) prediction is followed by a search which establishes the 'best' match MV with respect to a previously encoded frame. This function requires no modification to execute in a multi-processor environment since these frames have been encoded and are available to all processors. In addition, this process is executed more efficiently by vector/SIMD architectures due to the abundant DLP.

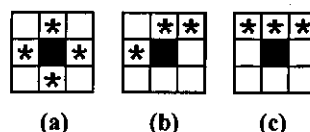


Figure 3: Ideal prediction, current prediction and proposed prediction

Unlike calculated MVs, predicted MVs are based on the current frame. Assuming that the motion transcends MB borders, the prediction function produces the average of the previously encoded MB's MVs. The timely availability of neighbouring MB MVs is different in a parallel implementation compared to the single-threaded case. Figure 3 shows three different prediction patterns that can be used to predict the motion of a specific MB. From the figure, it is clear

that the most accurate prediction is likely to come from taking the mean of all possible neighbouring MBs. Such a pattern is not possible since not all of the MBs required for the prediction have been previously encoded in either single or multi-threaded case. Using the scan line approach of figure 3b, the proposed method within MPEG-4 standard, MBs above and to the left can be used for predictions however, to allow for the MB in a row to be encoded in parallel, the block to the left cannot be used and the prediction pattern of figure 3c was thus implemented. A minimal fall in PSNR was observed when using this prediction pattern.

The MPEG-4 standard specifies that the differential between the calculated and predicted MV is stored in the bitstream. To allow the correct recovery of MVs from the later, the decoder must use the same predicted MVs. To ensure this once all MV for the row have been calculated, the original prediction algorithm is executed in serial and the differential calculated [9].

The block-based nature of the MPEG-4 encoder with the repetitive, independent computational steps involved leads naturally to parallelising the process at thread-level. Prior research in threading video workloads targeted a distributed network of workstations as the computation engine rather than a shared-memory multi-processor system [7]. In this case, the workload is distributed at far coarser granularity level (at the group of pictures, GOP, level) than our proposed technique since at this level a multi-computer architecture is more suitable due to the low inter-processor communication. However, such multi-computers are less capable to shared memory systems when implemented in SoC products due to the disjoint address space which calls for more silicon dedicated to private caches, per CPU. This is not the case for shared-memory configurations which can take the form of Chip-Multiprocessors (CMP), Multi-threaded processors (MT) or multi-threaded multi-processors (CMP-MT) in which much finer level of resource sharing can take place including the secondary cache subsystem (CMP/CMP-MT) or even the whole execution pipeline (MT/CMP-MT). We have therefore targeted such shared-memory systems due to their greater microarchitecture flexibility and potential for performance scalability and exploited TLP at a much finer granularity level the previous scheme.

Further to TLP, prior work by our group [8] as well as other researchers has focused on Data-Level-Parallelism (DLP) since this is the most widely accepted form of parallelism in media workloads. TLP exploitation, as discussed in the previous paragraph, yields orthogonal (thus, complementary) benefits to DLP allowing for seamless exploitation of both forms of parallelism. We believe that such a DLP-TLP

system is the optimal programmable solution for real-time video encoding SoCs.

5. Simulation Infrastructure

The threaded XviD encoder was compiled and verified on our custom, multi-context Instruction-Set-Simulator (MT-ISS) which is originally based on the SimpleScalar infrastructure [10]. This is a complete computer architecture research toolset and the ISS was extended to allow for arbitrary-large multi-context system modelling. The simulator produces dynamic instruction counts for a specified number of processor contexts and implements a software API to a cycle-accurate (CA) back-end. The combined MT-ISS and CA back-end will permit the detailed (near-RTL accuracy) study of arbitrary single and multi-processor configurations and interconnect.

6. Results

This section discusses the theoretical performance benefit of the threaded and vectorized XviD MPEG-4 video coder

6.1 TLP

Eight video sequences at CIF (352x288) resolution were encoded each consisting of 25 frames. The sequences were encoded for quality settings 1 through 5 and simulated on the MT-ISS for up to 64 processor contexts.

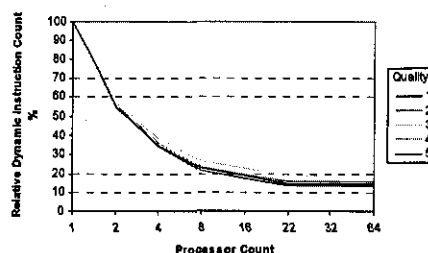


Figure 4 Foreman video sequence

Figure 4 depict the dynamic instruction count reduction for the primary processor context (thread 0) for the foreman video sequence. Context 0 is chosen as the reference because it is the controlling context as it performs I/O and activates the remaining contexts early during execution. The graph demonstrate a significant reduction in complexity metric with an increasing number of processors. No further complexity savings are achieved once the number of processor contexts exceeds, for the chosen video sequences, 22. The

reasoning behind this is that for maximum gain, the threaded loops should be executed once only which is true when the number of processor contexts is greater than or equal to the upper limit of the loop iterations. For a processor count less than this value a threaded loop is executed multiple times thus, leading to a larger instruction count per context. The optimal number of contexts for the MPEG-4 XviD workload is calculated by:

$$Opt_context(MPEG-4) = frame_width / 16 \quad [III]$$

For all three threaded functions the upper limit of the loop is directly related to the width of the input frames as specified above. This is imposed by the way the *MotionEstimation* function has been threaded and dominates the optimal number of threads of the other parallelised functions. To increase this limit a different implementation would need to be used since the data dependencies in the MV prediction are resolved for each row separately.

6.2 DLP

Functions of major complexity were identified during algorithm profiling as being ME, MC, DCT and IDCT and Q. The dynamic instruction count of these functions accounts for more than 80% of total complexity and the existence of significant amounts of data level parallelism makes them excellent candidates for vectorization [8]. Figure 5 shows the dynamic instruction count reduction, due to varying the architecturally-visible vector register length from 8 to 200 bytes, with results plotted for 3 quality settings. It shows an overall dynamic instruction count reduction of the order of 70% observed at a vector length of 32-bytes. These benefits relate to the single-threaded version of the workload however, they are orthogonal to the benefits achieved by exploiting TLP and are expected to remain largely insensitive, across all software threads.

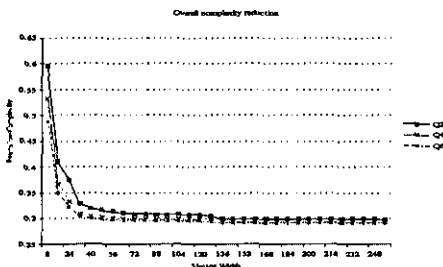


Figure 5: Overall dynamic instruction count reduction

7. VLSI Macrocells

DLP Accelerator

Following the theoretical study of the previous sections, we implemented a subset of the MPEG-4 vector ISA in the form of a custom tightly-coupled accelerator that attaches to an open-source, configurable, extensible, 32-bit RISC architecture [11] implementing the Sparc V8 [12] instructions et architecture (ISA).

The design has been previously presented in [13] and this section presents the VLSI implementation data of a more up-to-date implementation of the microarchitecture in which a number of critical paths were improved and more MPEG-4 related vector instructions added.

Figure 6a depicts the VLSI macrocell of the accelerator in a high-performance, 0.13 μm silicon process from UMC. The leftmost RAM cells are the 5-read, 1-write (5R1W) vector register file with geometry of 8x128 bit. The large memory arrays on top and bottom of the standard cells are the two banks of the local memory of the vector load-store unit (VLSU). Table 1 details the performance and characteristics of the macrocell.

Table 1: MPEG-4 VLSI Accelerator Characteristics

Parameter	Value
Std cells	25070
RAMs	8
Fmax	292.3 MHz
Size	1461 x 760 μm^2 (1111233 μm^2)

Cache-coherent multi-processor

We implemented a configurable, extensible, cache-coherent multi-processor based on the opensource CPU and internally developed IP. The implementation includes dual, modified Leon-2 CPUs each with an attached accelerator, as discussed above, in a shared-bus configuration. The VLSI layout and macrocell characteristics are presented in figure 6b and Table 2 respectively.

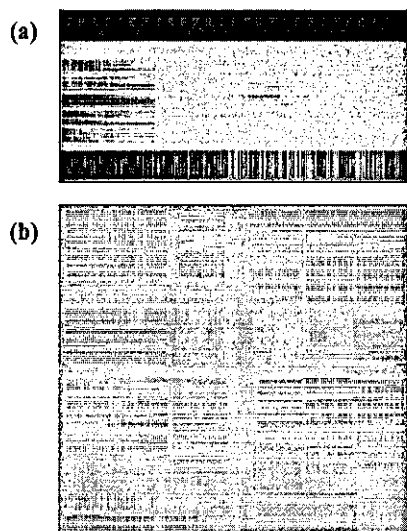


Figure 6: 2-way vector multi-processor system

Table 2: Cache-coherent multi-processors Characteristics

Parameter	Value
Std cells	83455
RAMs	52
Fmax	170 MHz
Size	2730 x 2732 μm^2 (7466115 μm^2)

Though the standalone accelerator achieves close to 300 MHz worst-case, post-route performance, this is reduced when it is attached to each of the two Leon-2 CPUs of figure 6. In the multiprocessor case, the critical path is in the shared bus infrastructure, significantly limiting the performance of the SoC multi-processor to 170 MHz whereas the accelerator has its critical path in the add-reduction logic of the SAD functionality. These results clearly indicate the potential of the standalone accelerator and the limitations, in terms of maximum operating frequency, of shared-bus multiprocessors.

8. Conclusions

This work quantified the Thread and Data-Level parallelism of an integer-only, open-source implementation of the high performance MPEG-4 video coding standard. Our results conclusively demonstrate that programmable solutions for consumer-driven products should address both forms of parallelism in order to achieve real-time video encoding, within the power budget of a consumer-market SoCs. Further work by our research groups will focus on fusing the single-thread, vectorized MPEG-4 encoder to the non-vectorized, multi-threaded version in order to present conclusively the benefit of tapping

the two most profound forms of parallelism, TLP and DLP. In addition, we shall be investigating cycle-effects of the proposed bus-based configuration and study more advanced interconnects.

8. Reference

- [1] MPEG1, ISO/IEC 11172-2:1993
- [2] Motion Picture Experts Group <http://www.mpeg.org>
- [3] MPEG Video Group, "MPEG-4 Video Verification Model 6.0", Doc. ISO/IEC JTC1 / SC29 / WG11 N1582, Sevilla MPEG Meeting, February 1997
- [4] G.J.Sullivan, P.Topiwala, A.Luthra, "The H.264/AVC Advanced Video Coding standard: overview and introduction to the Fidelity Range Extensions", SPIE conference on Applications of Digital Image Processing XXVII, August 2004.
- [5] Ax36 Family of Parallel Image and Video Digital Signal Processors DSP Chips', White Paper, Oxford Micro Devices, Inc, Monroe, CT 06468, USA, 2002
- [6] XviD core library source code, www.xvid.org
- [7] Ke Shen, Lawrence A. Rowe, and Edward J. Delp. A parallel implementation of an MPEG1 encoder: Faster than real-time! In Proceedings of SPIE Conference on Digital Video Compression: Algorithms and Technologies, San Jose, February 5-10 1995
- [8] V. A. Chouliaras, J. L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso 'A multi-standard video coding accelerator based on a vector architecture', Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2005), Las Vegas, Nevada, USA
- [9] Communication with C.Lambert, R.Czyz, XviD core developers. XviD-devel mailing list Sept.-Nov. 2004
- [10] SimpleScalar LLC <http://www.simplescalar.com/>
- [11] "The Leon-2 processor User's manual, XST edition, ver. 1.0.14", <http://www.gaisler.com>
- [12] The Sparc Architecture Manual Version 8', <http://www.sparc.org>
- [13] V. A. Chouliaras, J. L. Nunez-Yanez, S. Agha, 'Silicon Implementation of a Parametric Vector Datapath for real-time MPEG2 encoding', Proceedings of the IASTED (SIP) 2004, Honolulu, Hawaii, USA, ISBN: 0-88986-442-X

Paper PC9: S. R. Parr, K. Koutsomyti, V. A. Chouliaras, J.L. Nunez, D. J. Mulvaney, '*Configurable Scalar and Vector Coprocessors for accelerating the G.723.1 and G.729.A speech coders*', accepted for oral presentation at the IASTED International Conference on Signal and Image Processing (ACIT-SIP), Novosibirsk, Russia, June 20-24, 2005

CONFIGURABLE SCALAR AND VECTOR COPROCESSORS FOR ACCELERATING THE G.723.1 AND G.729A SPEECH CODERS

S. R. Parr, K. Koutsomyti, V. A. Chouliaras, J.L. Nunez, D. J. Mulvaney

Loughborough University

United Kingdom

s.r.parr@lboro.ac.uk

ABSTRACT

This paper presents the results of an investigation of employing configurable scalar and vector coprocessors to accelerate the G.723.1 and the G.729A speech coders. Architecture exploration has produced a reduction by up to 70% of the total number of instructions executed following the introduction of custom instructions. The accelerators are designed to be attached to a configurable embedded RISC CPU where they will make use of the host register file and load/store infrastructure.

KEY WORDS

Signal Processing, Coprocessor, Embedded systems, Speech coding.

1 Introduction

Speech compression is utilized in a multitude of communication applications[1][2][3], including Voice over Internet Protocols networks and digital satellite systems. Typical consumer products employing this technology are multimedia terminals, digital dictation machines, videophones and IP phones. The G.723.1[4] and the G.729A[5] recommendations were designed to standardize telephony and videoconferencing over public telephone lines and are part of the International Telecommunication Union (ITU) H.324 standard. This work investigates the benefit, in terms of complexity reduction, of architecture (instruction) extensions for the efficient execution of the above vocoders, building on previous work[6][7]. The identified extensions are implemented as coprocessors, tightly-coupled to a configurable, embedded RISC processor.

There is a significant body of research into application acceleration via targeted coprocessors; application domains are diverse, ranging from cryptography[8], maze-routing[9] to high-end video processing[10]. Previous research into efficient execution of speech coders include that by Costinescu *et al.*[11] and by Chang and Hu[12] which describe the necessary changes in the ITU reference code when targeting very high-performance, off-the-shelf digital signal processors. Soler *et al.*[13] describe a semi-automated chip-synthesis flow targeting a horizontally micro-programmed (VLIW) embedded DSP architecture, capable of executing one multiply-accumulate operation per clock cycle. The workload in this case was the GSM half-rate speech coder.

The research is a continuation of Raab *et al.*[10] which describes instruction set extensions, implemented in a moderate-complexity datapath (coprocessor) attached to a configurable embedded processor.

2 LPAS- Based Speech Coders

The G.723.1 and the G.729A standard speech coding algorithms, as recommended by the ITU, belong to the category of linear-prediction analysis-by-synthesis (LPAS) speech coders[14]. G.729A is a reduced complexity 8kbits/s version of the Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP) coder in the G.729 recommendation[5]. The G.723.1 dual rate speech coder for multimedia applications transmits at either 5.3kbits/s or 6.3kbits/s. Such coding schemes have been widely adopted as they produce high quality speech while maintaining a low bit-rate, but at the price of higher complexity.

The quality of speech improves with higher bit rates although the overall performance of the G.723.1 at 6.3kbits/s and the G.729A are similar. A clear difference in the performance of these two vocoders is their algorithmic delay; the total one-way delay of 25ms for G.729A compares favourably with that of 67.5ms for G.723.1. Technically, G.723.1 at 6.3kbits/s differs from the G.723.1 at 5.3kbits/s in the excitation model for the synthesis filter. The G.723.1 at 5.3kbits/s uses multi-pulse excitation with a maximum likelihood quantizer model while the G.723.1 at 6.3kbits/s and the G.729A uses the code excited linear predication model.

3 Problem Formulation

This research identifies architecture and microarchitecture requirements for the efficient implementation of the G.729A and G.723.1 speech coders on high-performance, low-cost, configurable microprocessors.

The workloads were executed and profiled in native mode (Linux x86). Table 1 shows the relative time spent outside the digital signal processor (DSP) emulation instructions. To research the potential acceleration of the algorithms when executed on an embedded microprocessor, the workload was recompiled for the SimpleScalar instruction set architecture (ISA). Table 2 illustrates the simulated

processor profiling results in terms of the number of instructions executed.

It is clear that the workloads spend a significant proportion of their time executing DSP emulation functions. If the DSP emulation instructions could be executed by configurable extensible microprocessor there is the potential to achieve a valuable reduction in execution time. A suitable high-performance, targeted-architecture for executing the workloads could reduce the form-factor and power consumption, making it a very attractive candidate for replication and integration in a System-on-Chip (SoC) ASIC.

Table 1: Relative amount of time spent outside the DSP emulation instructions

Algorithm	Relative time (% , native)
G723 Coder	31.3
G729 Coder	30.4

Table 2: Relative number of total instructions executed outside the DSP emulation instructions

Algorithm	Relative instructions (% , simulated)
G723 Coder	34.5
G729 Coder	34.2

4 Programmers Model

The programmer's model for the vector and scalar coprocessor accelerator is depicted in Figure 1. There are 16 vector registers (VR0-VR15), each consisting of a parametric number (VLMAX) of scalar (16-bit) elements. There are two vector accumulators (VACC0, VACC1) consisting of VLMAX/2 scalar elements (32-bits) and two vector mask registers of length VLMAX bits. Finally, there are 16 scalar registers and a sticky overflow flag.

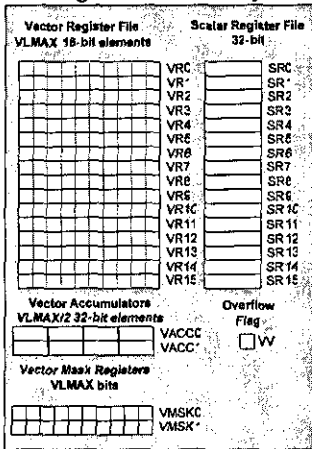


Figure 1: Scalar and vector accelerator programmers' model

5 Microarchitecture

The scalar and vector coprocessors are attached to the Sparc-v8 compliant CPU core via a custom pipeline coprocessor port[15][16]. For performance reasons, this approach was chosen in preference to designing an AMBA high speed bus (AHB) compliant master[17].

High-level views of both scalar and vector microarchitecture are depicted in Figure 3 and Figure 4 respectively.

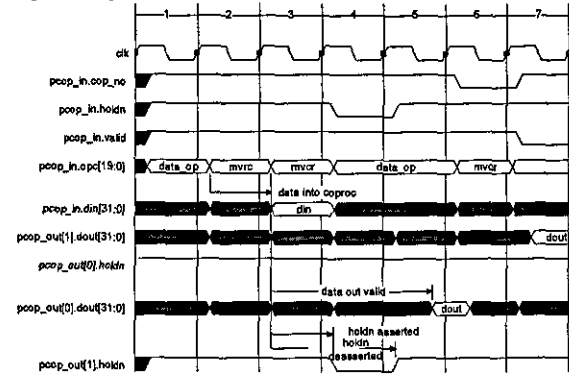


Figure 2: Typical Coprocessor Transaction

Typical read-write transaction of a coprocessor is depicted in Figure 2. The diagram shows a coprocessor data operation on cycle 1 followed by a host-to-coprocessor register transfer on cycle 2. In cycle 3, a coprocessor register is requested by the RISC processor but due to internal stall conditions, data is made available one cycle later than the expected time (cycle 5 instead of cycle 4). During that time, the main processor is held with the 'holdn' signal. Finally, a second read operation, this time directed to another coprocessor, is initiated in cycle 6. Results are made available to the main pipeline in cycle 7.

5.1 Scalar Microprocessor Architecture

This microarchitecture uses its own 16x32-bit register file. The coprocessor state is fully accessible from the RISC CPU. Bi-directional transfer instructions have been added between the host RISC processor and the coprocessors as both move-to-coprocessor and move-from-coprocessor instructions are absent in the Sparc v8 architecture.

The coprocessor pipeline is segmented into three main sections: Front-End, Control Pipeline and Datapath. The front-end reads instructions from the main CPU instruction cache and clocks them into the instruction register. The command is then decoded at the RISC processor and coprocessor and the read addresses are extracted. In parallel, the coprocessor computes the control fields used in its pipeline.

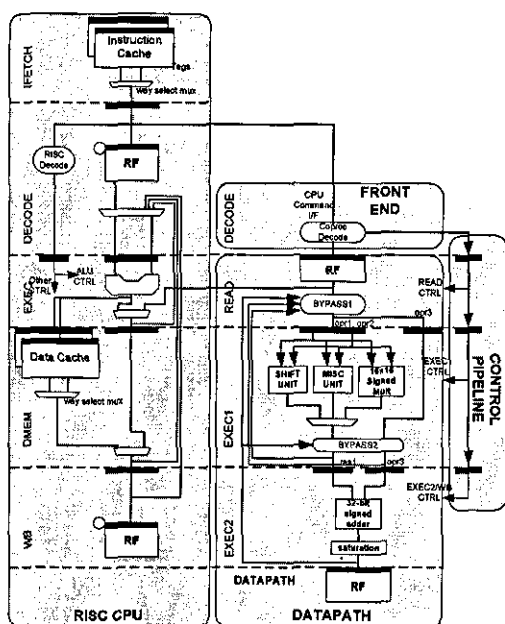


Figure 3: High-level scalar microarchitecture

5.1.1 EXEC1 Stage

EXEC1 includes datapath logic to perform 16x16 bit signed multiplication, all ITU shift operations and a miscellaneous block responsible for handling all opcodes not falling in other function blocks.

5.1.2 EXEC2 Stage

The results are passed from EXEC1 stage to this stage. Here, the add/sub part of the multiply-add or multiply-sub instruction is performed as well as the arithmetic and the saturation. Results commit to the private register file at the end of this cycle or return to the host pipeline during stage DMEM.

5.2 Vector Microprocessor Architecture

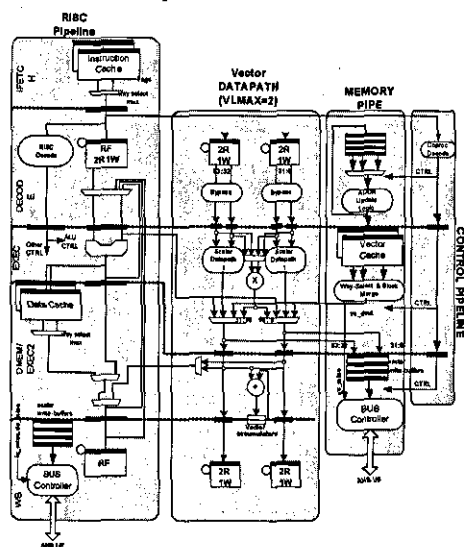


Figure 4: High-level vector microarchitecture

The vector coprocessor microarchitecture can handle both scalar and vector operations. The coprocessor consists of the parametric vector data path, the memory pipeline common to both scalar and vector coprocessors and the control pipeline. Within the vector data path, the access to the vector register file takes place at the same stage as access to the RISC processor register file, followed by bypassing of the vector operands. There are two stages to vector execution. In stage 1, all single-cycle operations are performed as well as the multiplication stage of the multiply-add/sub instructions. The second stage executes the addition/subtraction part of the multiply-add/sub along with reduction operations of the vector accumulators. Results are committed either to the vector accumulators or to staging registers prior to being written to the vector register file. This asymmetry is due to the long set up times of the vector register file dual-port random access memory (RAM).

The vector memory pipeline supplies scalar and vector operands to the scalar and vector accelerators respectively. Following scalar register file reads, the results are bypassed prior to an address being presented to the vector data cache. In the subsequent cycle, the vector data cache is accessed and the vector operands are returned to the vector pipeline. The vector store operations commit data to the vector write buffers (the vector data-cache is write-through) which sends a request to the AHB controller for write access to the bus. The bus controller is also used for data cache refill operations. The control pipeline generates and distributes all control signals to all stages of the vector accelerator.

6 System Architecture

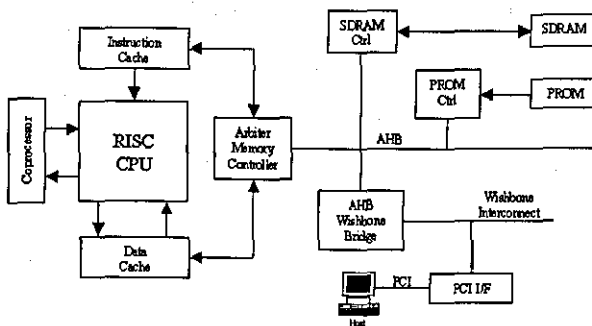


Figure 5: Overall System Architecture

The overall system architecture is shown in Figure 5. The combined CPU scalar and vector accelerators are connected to the system AHB bus which communicates with the external synchronous dynamic RAM (SDRAM) through an AHB/slave SDRAM controller. Speech frames to be processed are transferred from the host system via PCI interface which connects to the SoC kernel via a Wishbone Bridge.

The optimised speech coder and the frames to be processed are transferred with direct memory access

(DMA) from the host PC to the SDRAM memory of the RISC/coprocessor FPGA board, and this combination processes the frames and stores the compressed frames in local memory (SDRAM). The compressed frames are transferred back to the PC memory for comparison with the ITU-T test vectors.

7 Results

Results were obtained for both coprocessors at the architectural level, with the baseline architecture being the Simplescalar ISA. The workloads were compiled and all ITU test vectors were validated on the standard architecture simulator (sim-profile). Table 3 and Table 4 depict the number of simulated processor instructions required for each workload, for the G723.1 and G729A algorithms respectively.

Table 3: G723.1 unmodified instruction count

Test Vector	Coder	Instruction Count
dtx53mix.tin (Rate 5.3kbits/s)		925,853,310
dtx63.tin (Rate 6.3kbits/s)		10,159,685,901
dtx53mix.tin (Mixed Rate)		1,062,686,809

Table 4: G729A unmodified instruction count

Test vector	Instructions
	Coder
Alghm	62,613,675
Fixed	213,961,885
Lsp	3,977,183,504
Pitch	3,253,175,471
Tame	230,917,008
Test	311,692,276
Speech	6,656,625,331

The workloads were then modified to include custom assembly instructions and a new architecture-level simulator (sim-coproc), based on the existing profiling simulator, was designed. The test vectors were again simulated and the algorithmic complexity was measured and compared to that obtained in the previous run. Full compliance to the ITU-T test vectors was maintained throughout.

7.1 Vector Coprocessor Results

Figure 6 and Figure 7 show the results for the G.723.1 and G.729A respectively when a vector coprocessor was attached to the RISC processor. The complexities for both vocoders do not reduce further for vector lengths greater than 16-bits. The G.723.1 has a reduced complexity around 63% whereas the G.729A has a reduced complexity of around 55%.

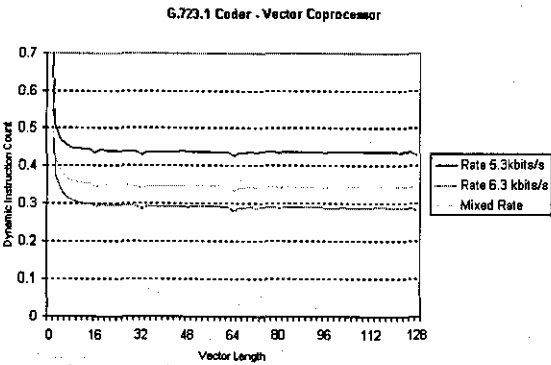


Figure 6 - G.723.1 Vector Coprocessor Results

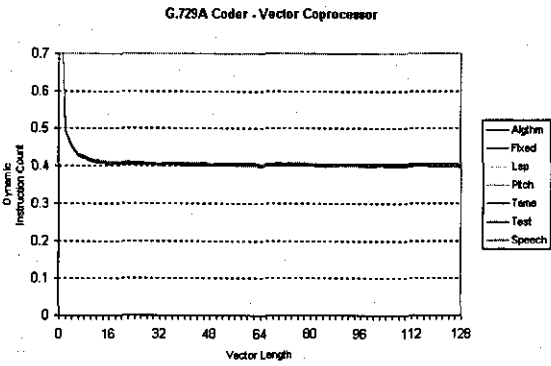


Figure 7 - G.729A Vector Coprocessor Results

7.2 Vector and Scalar Coprocessor Results

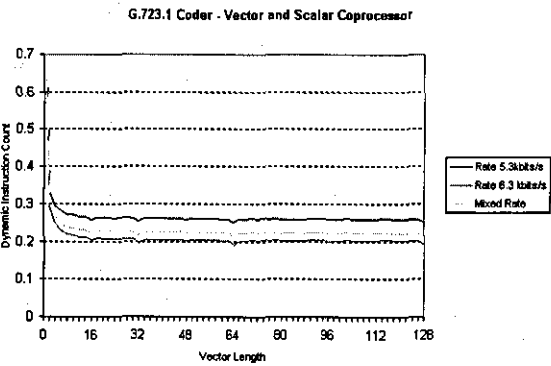


Figure 8 - G.723.1 Vector and Scalar Coprocessor Results

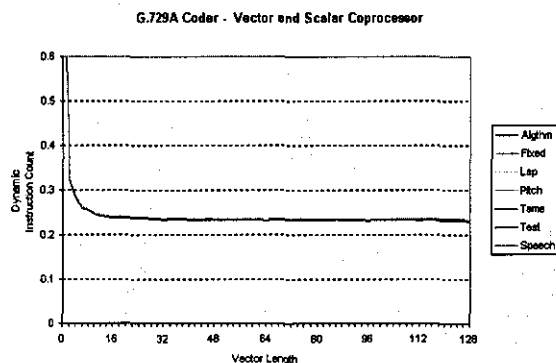


Figure 9 - G.729A Vector and Scalar Coprocessor Results

With the introduction of the scalar coprocessor, a significant improvement can now be seen. The G.723.1 now has a reduced complexity of 70% where as the G.729A has a reduced complexity of around 65%.

8 Conclusions and Future Work

The ITU-T G.729A and G.723.1 speech coders have been optimised by introducing custom scalar and vector instructions.

The results have shown a reduced complexity of 70% for the G.723.1 and 65% for the G.729A when both vector and scalar coprocessors are attached to the RISC CPU. Additional insight on the cycle effects of the combined scalar-vector architecture will be provided through cycle-accurate modelling of both coprocessors. This will allow for experimentation of the processor/co-processor design space and provide insight into necessary microarchitecture requirements for the efficient execution of the workloads.

The final stage of the research will be to build the register transistor logic (RTL) model of the vector and scalar coprocessors.

References:

- [1] Kondo, A.M (Ahmet M), *Digital Speech: coding for low bit rate communication systems* (New York, Chichester, Wiley, 1999).
- [2] R.Cox, P.Kroon, Low bit-rate speech coders for multimedia communication. *IEEE Communications magazine*, pp.24-41, December 1996
- [3] R Cox, Three New Speech Coders from the ITU cover a Range of Applications. *IEEE Communications Magazine*, vol.35, no.9, pp.40-47, September 1997
- [4] ITU-T Recommendation G.723.1, Dual Rate Speech coder for multimedia communications transmitting at 5.3 and 6.3 kbits/s. 3/96 www.itu.int
- [5] ITU-T Recommendation G.729, Coding of speech at 8 kbits/s using conjugate-structure algebraic-code-excited linear-prediction (CS-ACELP). 3/96 www.itu.int
- [6] V. A. Chouliaras, J. L. Nunez, A scalar coprocessor for accelerating the G723.1 and G729A speech coders. *IEEE Transactions on Consumer Electronics* vol.49, no.3, pp.703-710, Aug.2003
- [7] V. A. Chouliaras, J.L. Nunez, K. Koutsomyti, S.R. Parr, D.J. Mulvaney and S. Datta, Development of custom vector accelerator for high-performance speech coding. *IEE Electronics Letters*, vol. 40, no. 24, pp. 1559 – 1561, Nov. 2004
- [8] A. Royo, J. Moran, C. Lopez, Design and implementation of a coprocessor for cryptography applications. *Proceedings of the 1997 IEEE European Design and Test Conference (ED&TC'97)*, pg 213-217
- [9] Y. Won, S. Sahni, Y. El-Ziq, A hardware accelerator for maze routing. *IEEE Trans on Computers*, vol. 39, no. 1, pp. 141-145, Jan. 1990
- [10] A. Raab, N. Bruels, U. Hachmann, J. Harnisch, U. Ramacher, C. Sauer, A. Techmer, A 100-GOPS programmable processor for vehicle vision systems. *IEEE Design and Test of Computers*, pp.8-16, Jan-Feb 2003
- [11] B. Costinescu, R. Ungureanu, M. Stoica, E. Medve, R. Pread, M. Alexiu, C. Ilas, ITU-T G729 Implementation on Starcore SC140. *AN2094/D*, Rev. 0,02/2001, www.motorola.com
- [12] S. Chang, J. Hu, Real-time implementation of G723.1 speech codec on a 16-bit DSP processor. *Department of electronic and control engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C*
- [13] M. Soler, A. Andre, E. Closse, J. Laval, F. Balestro, D. Morche, P. Senn, An embedded DSP platform for multi-standard ITU G728, G729 & G723.1 audio compression. *France Telecom, CNET*
- [14] A S Spanias, Speech Coding: A Tutorial Review. *Proceedings from the IEEE* vol. 82, no. 10, pp.1541-1581, October 1994
- [15] The Leon-2 processor User's manual, XST edition, ver. 1.0.14. www.gaister.com
- [16] The Sparc Architecture Manual Version 8. www.sparc.com
- [17] AMBA Specification (Rev 2.0). www.arm.com

GROUP RJ: Under-review Journal papers

This final set of journal contributions includes papers that are still under review and as such, they are mentioned in this section only for completeness:

RJ1:

José L. Núñez, V. A. Chouliaras, '*A 1 Gigabyte per Second Streaming Lossless Data Compression Chip Based on a Configurable Variable-Geometry CAM Dictionary*', submitted to IEE Proceedings on Computer and Digital Techniques, June 2004. Accepted for publication.

This paper discusses the VLSI implementation of a streaming, high performance, CAM-based, lossless data compressor capable of processing 4 input bytes per cycle. The resulting macrocell occupied a silicon area of 2 mm by 1 mm and achieved a maximum post-route frequency of 273 MHz in a high performance CMOS process resulting in a compression bandwidth in excess of 1 GB/s. This architecture was the Ph.D. outcome of my colleague, Dr. Jose Nunez. My contribution was the fine-tuning of the design RTL code for synthesis by the Synopsys tools and the subsequent back-end flow. This contribution has been accepted for publication.

RJ2:

Xiaofeng Wu, V. A. Chouliaras, Jose Nunez- Yanez and Roger Goodall, '*On the design of a $\Delta\Sigma$ control system processor VLSI hard macro*', submitted to IEEE Transactions on Very Large Scale Integration, October 2004. Accepted for publication.

In their quest for ultimate performance, industry and academia have been researching and developing high performance DSP engines which can process a number of binary words every cycle. This paper proposes a very interesting alternative in which a bit-serial DSP architecture for control applications is proposed by my colleagues, Prof. Roger Goodall and Mr. Xiaofeng Wu. The bit-serial approach delivers *sufficient* processing capability to achieve the required sampling rates and at the same time, dispenses with the use of bit-parallel multipliers. My responsibility in this work was in the preparation of the RTL for front-end synthesis and the whole of the back-end flow, for two configurations of the program RAM of the design. This contribution has been accepted for publication.

RJ3:

V. A. Chouliaras, J. L. Nunez, D. J. Mulvaney, '*On the systematic development of a parametric vector accelerator for high performance video coding*', submitted to IEEE Transactions on Very Large Scale Integration, October 2004

This contribution goes into great length at discussing the parallelization (vectorization) process of the reference MPEG-2 application software. Starting with the reference application, the code is profiled and the major CPU cycle consumers are identified at the function level. Visual inspection revealed that these computationally-intensive functions have sufficient DLP which was exploited via the introduction of a custom vector ISA extensions, leading to a maximum dynamic instruction count reduction of the order of 68%. My responsibility in this work was in the architecture specification, implementation, and simulation of the vector instruction set architecture. I subsequently was the main designer behind the implementation of the MPEG-2 ME accelerator and the combined processor-coprocessor VLSI macrocell.

RJ4:

V. Dwyer, S. Agha, V. A. Chouliaras, '*Motion estimation with low resolution distortion metric*', submitted to IEE Electronic Letters, Jan 2005. Accepted for publication.

This work elaborates quite substantially on the Reduced Bit-width SAD (RBSAD) methodology as discussed in paper PC7. It has been accepted for publication.

RJ5:

J. L. Nunez, V. A. Chouliaras, '*High-Throughput Arithmetic Coding Hardware for the H264 Advanced Video Compressor*', submitted to IEEE Transactions on Circuits and Systems for Video Technology, February 2004, revision stage 1

This contribution proposes a high-performance, fully-pipelined (1 symbol per cycle), binary arithmetic coding architecture that replaces multiplication and division operations by look-up table probing. The microarchitecture of the encoder and the decoder consists of 5 and 2 stages respectively and achieves a throughput of 70 MSymbols/s on an FPGA or 315 MSymbols/s on a high-performance 0.13 μm CMOS process. A major characteristic of the microarchitecture is the renormalization operation which takes place in one cycle thus permitting the processing of one symbol per cycle.

RJ6:

J.L. Núñez, V. A. Chouliaras, '*A Configurable Statistical Lossless Compression Core Based on Variable Order Markov Modelling and Arithmetic Coding*', submitted to IEEE Transactions on Computers, October 2004. Accepted for publication.

The Electronic Systems Design Group has been very active in the research of streaming, lossless data compression technologies over a number of years. In particular, the group has been quite successful in the development of a dictionary-based, streaming compressor/decompressor architecture known as XMatchPRO which demonstrated on average, a 50% to 60% compression ratio on established benchmarks. This work takes a radical departure from dictionary-based approaches by discussing the first hardware implementation of a configurable statistical lossless compression engine based on variable-order Markov modelling and arithmetic coding. The engine is known as Byacom-1.

Appendix A: Complete list of Author Publications

A.1 Academic Journal Publications (published)

1. V. A. Chouliaras, J. L. Nunez, '*Scalar Coprocessors for accelerating the G723.1 and G729A Speech Coders*', IEEE Transactions on Consumer Electronics, Vol. 49, Issue 3, Aug. 2003, pg. 703-710, ISSN:0098-3063.
2. Grecos, C., Saparon, A. and Chouliaras, V., '*Three novel low complexity scanning orders for MPEG-2 full search motion estimation*', Real Time Imaging, 10, February 2004, pp 53-65, ISBN 1077-2014
3. V. A. Chouliaras, J. L. Nunez, K. Koutsomyti, S. R. Parr, D. J. Mulvaney, S. Datta, '*On the development of a custom vector accelerator for high-performance speech coding*', IEE Electronic Letters, Vol. 40, Issue 24, 25 Nov. 2004, pg 1559-1561, ISSN 0013-5194
4. V. A. Chouliaras J. L. Nunez, D. J. Mulvaney, F. Rovati, D. Alfonso, '*A Multi-standard Video coding accelerator based on a vector architecture*', IEEE Transactions on Consumer Electronics, Vol. 51, Issue 1, Feb 2005, pg 160-167, ISSN:0098-3063
5. J. L. Nunez, V. A. Chouliaras, '*High Performance Arithmetic Coding VLSI Macro for the H264 Video Compression Standard*', IEEE Transactions on Consumer Electronics Vol. 51, Issue 1, Feb 2005, pg 144-151 ISSN:0098-3063 Academic Journal Publications (under review)

A.2 Academic Journal Publications (under review/accepted for publication)

1. José L. Núñez, V. A. Chouliaras, D. J. Mulvaney, '*A 1 Gigabyte per Second Streaming Lossless Data Compression Chip Based on a Configurable Variable-Geometry CAM Dictionary*', submitted to IEE Proceedings on Computer and Digital Techniques, June 2004. This work has been accepted for publication.
2. Xiaofeng Wu, V. A. Chouliaras, Jose Nunez- Yanez and Roger Goodall, '*On the design of a $\Delta\Sigma$ control system processor VLSI hard macro*', submitted to IEEE Transactions on Very Large Scale Integration, October 2004. This work has been accepted for publication.

3. V. Dwyer, S. Agha, **V. A. Chouliaras**, '*Motion estimation with low resolution distortion metric*', submitted to IEE Electronic Letters, Jan 2005. This work has been accepted for publication.
4. E. Touloupis, J. A. Flint, **V. A. Chouliaras** and D. D. Ward, '*Modelling Multiple Faults in Fault-Tolerant Processor Architectures*', submitted to IEE Electronic Letters, February 2004.
5. J. L. Nunez, **V. A. Chouliaras**, '*High-Throughput Arithmetic Coding Hardware for the H264 Advanced Video Compressor*', submitted to IEEE Transactions on Circuits and Systems for Video Technology, February 2004, revision stage 1
6. J.L. Núñez, **V. A. Chouliaras**, '*A Configurable Statistical Lossless Compression Core Based on Variable Order Markov Modelling and Arithmetic Coding*', submitted to IEEE Transactions on Computers, October 2004. This work has been accepted for publication.

A.3 Refereed Conference Publications (presented/accepted)

1. **V. A. Chouliaras**, '*A new approach in the integrated automation of pharmacies based on novel concepts of functional and ergonomical design*', proceedings, 1st National Pharmaceutical Conference, June 1994, Athens, Greece.
2. **V. A. Chouliaras**, D. E. Edwards, '*A Superscalar AMULET*', proceedings, 1st UK Async Forum, December 1996, Edinburgh, Scotland
<http://www.dcs.ed.ac.uk/home/cxs/forum.html>.
3. **V. A. Chouliaras**, J. L. Nunez, '*A Scalar Coprocessor for accelerating the G723.1 and G729A Speech Coders*', Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2003), Los Angeles, California, USA, ISBN:0-7803-8838-0.
4. Nikolova, E.G.; Mulvaney, D.J.; **Chouliaras, V.A.**; Nunez-Yanez, J.L., '*A code compression scheme for improving SoC performance*', Proceedings of the 2003 IEEE International Symposium on System-on-Chip, IEEE Cat. No.03EX748.
5. E. G. Nikolova, D. J. Mulvaney, **V. A. Chouliaras**, J.L. Núñez, '*A Novel Code Compression/Decompression Approach for High-performance SoC Design*', Proceedings of the IEE Seminar on SoC Design, Test and Technology, Cardiff University, Cardiff, UK, 2 September 2003.
6. **Chouliaras, V.A.**, Flint, J.A. and Li, Y., '*Towards a Custom Vector-Parallel Machine for TLM*', Proceedings, Fifth IEE International Conference on Computation in

- Electromagnetics , IEE Conf Pub 505, IEE, CEM 2004, Stratford-upon-Avon UK, April 2004, 2004, 33-34, ISBN 086341 4001
7. Touloupis, E., Flint, J. A. and **Chouliaras, V. A.**, '*A Fault-Tolerant Architecture for Automotive Applications*' , Proceedings of PREP 2004, EPSRC, University of Hertfordshire UK, April 2004, 2004, 90-91.
 8. **V. A. Chouliaras**, J. L. Nunez-Yanez, S. Agha, '*Silicon Implementation of a Parametric Vector Datapath for real-time MPEG2 encoding*', Proceedings of the IASTED (SIP) 2004, Honolulu, Hawaii, USA, ISBN: 0-88986-442-X
 9. X. Wu, **V. A. Chouliaras**, R. M. Goodall, '*An application-specific processor hard macro for real-time control*', Proceedings of the IEEE SOCC 2004 Conference, San Jose, CA, USA
 10. J.L. Núñez, **V. A. Chouliaras**, '*Arithmetic Coding Hardware Acceleration in a SOPC Platform for Advanced Video Compression*', Proceedings, International Conference on Reconfigurable Computing and FPGAs (ReConFig04), pp. 19-28, ISBN 970692169-9, Colima, Mexico, September, 2004.
 11. **V. A. Chouliaras**, J. A. Flint, Y. Li, '*Parametric Data-Parallel architectures for TLM acceleration*', Proceedings of the 3rd International Conference on Computational Electromagnetics and Its Applications (ICCEA), Nov. 1-4 2004, Beijing, China, ISBN
 12. J. L. Nunez, **V. A. Chouliaras** '*High-performance Arithmetic Coding VLSI Macro for the H264 Video Compression Standard*', Proceedings of the IEEE International Conference on Consumer Electronics (ICCE 2005), Las Vegas, Nevada, USA, ISBN: 07803-8839-9
 13. **V. A. Chouliaras**, J. L. Nunez, Fabrizio. S. Rovati, Daniele Alfonso '*A multi-standard video coding accelerator based on a vector architecture*', Proceedings of the IEEE International Conference in Consumer Electronics (ICCE 2005), Las Vegas, Nevada, USA, ISBN: 07803-8839-9
 14. K. Koutsomyti, S. R. Parr, **V. A. Chouliaras**, J. Nunez, D. J. Mulvaney, S. Datta, '*Scalar and Parametric Vector Accelerators for the G.729A Speech Coding Standard*', Proceedings of IEE/ACM Postgraduate Seminar on Soc Design, Test and Technology , IEE , Loughborough University UK, 15th September 2004 , ISBN 0 86341 460 5
 15. E Touloupis, J A Flint, **V A Chouliaras** and D Ward, '*A TMR processor architecture for safety-critical automotive applications*', Proceedings of IEE/ACM Postgraduate Seminar on Soc Design, Test and Technology, IEE, Loughborough University UK, 15th September 2004, ISBN 0 86341 460 5

16. T. R. Jacobs, V. A. Chouliaras, D. J. Mulvaney, J. L. Nunez '*The application of Thread-Level Parallelism for reducing the architectural complexity of an MPEG-2 encoder*', Proceedings of IEE/ACM Postgraduate Seminar on Soc Design, Test and Technology, IEE, Loughborough University UK, 15th September 2004, ISBN 0 86341 460 5
17. E. Touloupis, J. A Flint, V. A. Chouliaras, D. D Ward, '*A Fault-tolerant Processor Core Architecture for Safety-Critical Automotive Applications*', Proceedings of the SAE 2005 World Congress & Exhibition, April 2005, Detroit, MI, USA (Document Number: 2005-01-0322)
18. K Koutsomyti, S R Parr, V A Chouliaras, J Nunez, D J Mulvaney, S Datta, '*Configurable Scalar and Vector Accelerators for the G.729A and G.723.1 Speech Coding Standards*', Proceedings of the EPSRC/IEEE/IEE Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science (PREP2005), University of Lancaster, UK, 30 March-1 April 2005
19. T. R. Jacobs, V A Chouliaras, D J Mulvaney, '*Investigation of Thread-Level Parallelism in the architectural complexity reduction of MPEG-2 and XViD video encoders*', Proceedings of the EPSRC/IEEE/IEE Postgraduate Research Conference in Electronics, Photonics, Communications and Networks, and Computing Science (PREP2005), University of Lancaster, UK, 30 March-1 April 2005
20. V M Dwyer, S Agha and V Chouliaras , '*Low power full search block matching using reduced bit SAD values for early termination*', Proceedings of the Mirage 2005 International conference on Computer Vision/Computer Graphics collaboration techniques, Paris, France
21. S. R. Parr, K. Koutsomyti, V. A. Chouliaras, J.L. Nunez, D. J. Mulvaney, '*Configurable Scalar and Vector Coprocessors for accelerating the G.723.1 and G.729.A speech coders*', accepted for oral presentation at the IASTED International Conference on Signal and Image Processing (ACIT-SIP), Novosibirsk, Russia , June 20-24, 2005
22. V. A. Chouliaras, Ashwin K. Kumaraswamy, T. R. Jacobs, and J. L. Nunez-Yanez, '*System-on-Chip Design Framework (SDF) unifying Specification Capture and Design Modelling*', Proceedings of the 2005 Electronic Design Processes (EDP) Workshop, April 6-8, Monterey Beach Hotel, Monterey, California, USA
23. Indrajit Atluri, Ashwin K. Kumaraswamy and V. A. Chouliaras, '*Energy efficient architectures for the Log-Map decoder through intelligent memory usage*', Proceedings of the IEEE Annual Symposium on VLSI, May 11-12 2005, Tampa, Florida, USA

24. **V. A. Chouliaras**, J. L. Nunez-Yanez, T. R. Jacobs and Ashwin K. Kumaraswamy, '*Configurable Multiprocessors for high-performance MPEG-4 video coding*', Proceedings of the IEEE Annual Symposium on VLSI, May 11-12 2005, Tampa, Florida, USA
25. Tom R. Jacobs, **V. A. Chouliaras** and Jose L. Nunez, '*A Thread and Data-Parallel MPEG-4 Video Encoder for a System-On-Chip Multiprocessor*', accepted for oral presentation at the IEEE 16th International Conference on application-specific architectures and processors (ASAP 2005), Samos, Greece, July 23-25 2005
26. José L. Núñez-Yáñez, **V. A. Chouliaras**, '*Design and Implementation of a High-Performance and Silicon Efficient Arithmetic Coding Accelerator for the H.264 Advanced Video Codec*', accepted for oral presentation at the IEEE 16th International Conference on application-specific architectures and processors (ASAP 2005), Samos, Greece,
27. **Vassilios A. Chouliaras**, Vincent M. Dwyer and Sharukh Agha, '*On the performance improvement of sub-sampling MPEG-2 Motion Estimation Algorithms with vector/SIMD architectures*', accepted for oral presentation at the 'Advanced Concepts for Intelligent Vision Systems' (ACIVS2005) conference, University of Antwerp, Antwerp, Belgium
28. Vincent M Dwyer, Sharukh Agha and **Vassilios A Chouliaras** , '*Reduced-Bit, Full Search Block-Matching Algorithms and their Hardware Realizations*', accepted for oral presentation at the 'Advanced Concepts for Intelligent Vision Systems' (ACIVS2005) conference, University of Antwerp, Antwerp, Belgium
29. K. Koutsomyti, S. R. Parr, **V. A. Chouliaras**, J. Nunez, '*Applying Data-Parallel and Scalar Optimizations for the efficient implementation of the G.729A and G.723.1 Speech Coding Standards*', accepted for oral presentation at the Seventh IASTED International Conference on Signal and Image Processing (SIP 2005), Honolulu, Hawaii, USA

A.4 Patent applications

The following two patent applications are in progress

1. D. J. Mulvaney, A. Kumaraswamy, **V. A. Chouliaras**: USPTO application for novel specification capture and electronic system design tool
2. E. Touloupis, J. Flint, **V. A. Chouliaras**, D. Ward: USPTO application for novel, fault-tolerant RISC CPU microarchitecture for mission critical systems

A.5 Non-Refereed Contributions

This section lists a number of non-refereed publications by the author and his research group

1. **V. A. Chouliaras**, D. Edwards, '*DDANIA: The Don't Dare Not Issue Architecture*', Amulet Group Internal Report, Department of Computer Science, University of Manchester, 1996.
2. E. G. Nikolova, D. J. Mulvaney, **V. A. Chouliaras**, J. L. Nunez-Yanez, '*A compression/decompression scheme for embedded system codes*', Proceedings, Electronics Systems and Control Division Mini-Conference, 25/9/2003, University of Loughborough
3. J. L. Nunez and **V. A. Chouliaras**, '*Variable-Order Context-Based Statistical Hardware Data Compression for Wireless Networks and Mobile Computing*', Proceedings, Electronics Systems and Control Division Mini-Conference, 25/9/2003, University of Loughborough
4. Z. M. Yusof, D. J. Mulvaney, J. L. Nunez-Yanez, **V. A. Chouliaras**, '*Compressed Memory Core*', Proceedings, Electronics Systems and Control Division Mini-Conference, 25/9/2003, University of Loughborough
5. Yibin Li, **V. A. Chouliaras**, James A. Flint and David J. Mulvaney, '*Exploration of Data and Thread Level parallelism in Transmission Line Modelling*', Proceedings, Electronics Systems and Control Division Mini-Conference, 25/9/2003, University of Loughborough
6. Tom Jacobs, **V. A. Chouliaras**, David Mulvaney, '*Investigation of Thread-Level Parallelism in the Architectural Complexity Reduction of MPEG2, XviD and H.264 Video Encoders*', Proceedings, Electronics Systems and Control Division Mini-conference, February 2005, University of Loughborough
7. S. R. Parr, K. Koutsomyti, **V. A. Chouliaras**, Jose Nunez, D. J. Mulvaney, S. Datta, '*Configurable Scalar and Vector Coprocessors for the G.723.1 and G.729A Speech Coders*', Proceedings, Electronics Systems and Control Division Mini-conference, February 2005, University of Loughborough

