Design Space Exploration of heterogeneous SoC Platforms for a Data-Dominant Application


Universitat Autònoma de Barcelona

ETSE Engineering Superior Technical School

PhD in Computer Science (Computer Engineering)

Antoni Portero Trujillo

Director: Prof. Jordi Carrabina Bordoll

Advisor: Prof. Francky Catthoor

08193 Bellaterra, Spain

June 22, 2009

" Wir lassen nie vom Suchen ab,
und doch, am Ende allen unseren Suchens
, sind wir amb ausgangspunkt zurück
und werden diesen Ort zum ersten Mal erfassen...

T.S. Eliot

...für Myrna Minkoff, Marla Singer, Judith Olofernes und Tofupax(2005/02/06) ..."

Acknowledgments

Thanks to my academic Director Dr. Jordi Carrabina, for giving me the opportunity to write this thesis and join the Microelectronics and Electronics Systems Department MiSE at the University Autònoma de Barcelona. Thanks a lot to Guillermo Talavera for investing in me his last two years and giving a push to my thesis work. Thanks to Joaquim Saiz, Dr. Lluís Ribas, and J.A. Velasco, because I cannot thank them enough for the opportunities and protect me and stay with me in the hard moments.

Thanks for the teaching people who I worked for during these long years, Raul Aragonès, Mercedes Rullan, Jordi Aguiló, Elena Valderrama and all the professors and teachers of the MiSE Department. I also want to thank the excellent Department staff for their support, Toni Guerra, and all the other people that is not here anymore but they're in my mind, Montse,... Thanks to all my colleagues at work, Oscar Navas, Ramon Pla, Marius, Aitor, Borja, Marc for their support and criticism, part of this work is theirs. Thanks to the CEPHIS team, David Castell, Eloi Ramon, Ramon, Enric, Marc, Jorge Luis, Pablo, Jordi, Victor, Andreu, Miquel, and Cesar.

Many thanks to Dr. Francky Catthoor for the guidance and for sharing his expert knowledge and helping me to take decisions of this thesis. I also want to thank the people I met at IMEC, Nacho, Pol, Yang ("Punk")...etc. Thanks to my sister Dolores for encouraging me to finish this thesis, her constant support throughput these years; she's always believed in me. I could not have reached this without her. Thanks to my dad and mum. I hope you will be proud of me, I am proud of you. Thanks to my brother Pedro, his wife Ana and my nieces Patricia and Aida. Thanks to the family that I met in Brussels, Fran, Asis, Joanma, Pierre (and his wife I hope) and all the people who were in the Samaritaine. I miss you a lot.

Thanks to the people that I met at Luton University, UK, Don Dent and Richard. Thanks for all proofreading help that Ramon Carrabina offered while I was writing this thesis. Thank you to my Adventures' colleagues, Clemente, Rodrigo, David, Jose, Dani ...and my current flat mates David, Miki and Pedro. Thank you to all the girls who took part of my hearth during these long years, Agata, Ana, Sonia, Vanina, Leatitia, Juliet, Janet, Julia...and so forth, so long... And to all the people that I have convinced to develop their final career projects with me.

Contents

Glossary of Terms

4CIF Four Common Intermediate Format (704x576 pixels)
AAC Advanced Audio Coding
ADC ADC Analog to Digital Converter / Adaptive Data Compression (MODEM) dict.die.net/adc/
API Application Program Interface
ASIC Application Specific Integrated Circuit
ASSP Application Specific Standard Product
CAE Computer Aided Engineering
CBR Constant Bit Rate
CCD Charge-Coupled Device
CMOS Complementary Metal Oxide Semiconductor
CIF Common Interface Format
CPLD Complex Programmable Logic Device
CPU Central Processing Unit
CSMA Carrier Sense Multiple Access
DAC Digital to Analog Converter
DMSoC Digital Media System-on-Chip
DCT Discrete Cosine Transform
DMA Direct Memory Access
DSP Digital Signal Processor
EDMA Enhanced DMA (EDMA) controller
ESL Electronic System Level
GDSII Graphic Design Station II
GPP sGeneral Purpose Processor
IDCT Inverse Discrete Cosine Transform
DLNA Digital Living Network Alliance
DSP Digital Signal Processor
DVB Digital Video Broadcasting
ECC Elliptic Curve Cryptography
EDIF stands for Electronic Design Interchange Format
FPGA Field Programmable Gate Array
FLI Foreign Language Interface
FSB Front Size Bus
FSM Finite State Machines Gbps Giga bits per second
GOB Group Of Pictures H.263
GPP General Purpose Processors
ITRS International Technology Roadmap for Semiconductors
ITU-R standard for digital video compression for videoconferencing. Uses the DCT "Video Coding for Low Bitrate Communication"
H.323 ITU-R standard that specifies the components, protocols and procedures that provide multimedia communication
HD Hard Disk

HDTV(16:9) High Definition Television
HQ High Quality
HVS Human Visual System
HW Hardware
IC Integrated Circuit
IEE Institution of Electrical Engineers
IEEE Institute of Electrical and Electronics Engineers
ILP Instruction Level Paralization
IMC Memory Controller
ISA Instruction Set Architecture
ISS Instruction Set Simulator
IP Intellectual Property
IPC Inter Process Communications
ISP Instruction Set Processor
ITU-R International Telecommunication Union - Radio communications Bureau
ITRS International Technology Road map for Semiconductors
JHVM Java Hardware Virtual Machine
JHDL Java Hardware Description Language
Kbps Kilo Bits Per Second
LAN Local-Area Network Mbps Millions of Bits per Second
MiB Mebibyte (a contraction of mega binary byte) is a unit of digital information storage. 1 MiB = 220 bytes = 1,048,576 bytes = 1,024 kibibytes
MIPS Instructions per Second
MoC Model of Computation
MOPS Operations per Second
MPSoC Multi-Processor Systems-on-a-Chip
NoC Networks-on-a-Chip
MPEG Moving Picture Experts Group
NP Network Processor
NTSC National Television System Committee
OMA Open Mobile Alliance
OSCI I Online Service Computer Interface
OSD On Screen Display
PAL Phase Alternating Line
PCB Printed Circuit Board
PCMCIA Personal Computer Memory Card International Association
PBD Platform-Based Design
PE Processor Element
PES packets Packetised Elementary Stream packets
PHY Physical Layer
PLL Phase-Locked Loop
PSNR Peak Signal to Noise Ratio
QCIF Quarter Common Interface Format
QoC Quantum of Computation

QoS Quality of Services
RF Radio Frequency
RGB Red Green Blue
RIC Reconfigurable Co-Processor
RSVP Resource Reservation Protocol
RTP Real Time Protocol
RVLCs Reversible Variable Length Codes
SERDES SERializer/DESerializer
SIMD Single Instruction Multiple Data
SLDL System-Level Design Language
SMP Symmetric Multi-Processing
SoC System on Chip
SPR Synthesis Placement Route
STB Set Top Boxes
STC System Time Clock
SW Software
TBB Threading Building Blocks
TDP Thermal Dissipation Power
TPL Task Level Parallelization
TS Transport Stream
UDP User Datagram Protocol
VCD Value Change Dump
VHS Video Home System
VLD Variable Length Decoding
VLE Variable Length Encoding
VBR Variable Bit Rat
VLIW Very Long Instruction Word
VHDL V (Very High Speed Integrated Circuit (VHSIC)) Hardware Definition Language
VHS Video Home System
VC Virtual Components
VSP Video Signal Processor
WCET Worst Case Execution Time
WLAN Wireless Local-Area Network
YCbCr The color space used in the CCIR-601 digital video specification
YUV Y=Luminance, U=Normalized BY, V=Normalized RY
VGA Video Graphics Array
W3C World Wide Web Consortium

1 Electronics Spectrum of Solutions: Architectures and Languages

Thesis Introduction

PC era is disappeared [1]. Currently innovative wireless devices interconnected are assaulting the marketplace. These gadgets will have to support very high flexibility of the Software and Reconfiguration of the Hardware to achieve the entire requirements. In the past decade, PC and Internet gave rise the .COM world. With the current progress of sub-100nm technology, DSP, MEMS technology, RF CMOS and nanoelectronics which rapidly merges with biotechnology, we are now entering the post PC and post .COMera. We are evolving towards a smart environment (ambient intelligence) in which we will be surrounded by smart things that communicate with each other and thereby enhance our consciousness, protect our health and globally connect people and things. The driven force will no longer be the PC but information technology that adapts to human being rather than vice versa. Designing such systems differs radically from designing CPU architectures. However, design technologists are confronted with giga scale complexity. The real bottleneck will not be in process technology itself but in our ability to design products and services into such huge multiprocessor architectures on a single piece of silicon or in a single package. The era of ambient intelligence or pervasive computing is based on devices that interact in an intelligent way with the human being. Multiprocessor architectures already exist from way back. The new challenge is now to make them adaptable to the human behavior. Therefore, they have to be made reconfigurable by software that is distributed over the network. This will cause a paradigm shift in the Computer Aided Engineering (CAE) world as embedded software will have to be developed together with the multi-processor architectures it will run on and this under the extreme-low energy constraints. Therefore, one of the major bottlenecks with energy consumptions in such kind of platforms is the data fetching from the different layer memories to the different processors [2]. Currently, most designs are written in software like C++, Matlab etc. without taking into account the optimal implementation. However, energy optimization can already start at a higher level of design without knowing the exact architecture. Instead of writing code as a sequence of functions calls, task-concurrency management, and distributed storage and computation can already result in a decrease in power consumption. Once the architecture is defined, further optimization can be performed. Until now, there are no CAD tools commercially available to make such power-efficient designs and a lot of research will be needed to develop such tools.

This thesis presents a detailed design methodology starting from a uniform reference model (MPEG compressor) and taking into account energy concerns that let to diverse

implementations of a complex IP targeted to heterogeneous platforms (such as: ASIC, FPGA, DSP plus an embedded processor, and an ASIP). This lets to the analysis of the obtained results facing mainly flexibility issues. Different improvements are made with platform independent and platform dependent optimizations. Besides, an accurate study at optimal work points balancing execution time vs. power trade off, depending on the number of functional units, has also been achieved . In addition a high-level functional NoC has been developed where this MPEG IP is connected as a NoC tile.

1.1 Electronic Systems Design

The complexity of electronic design is a powerful concept for coping with the increasing pressure on time to market, design and manufacturing cost. The complexity of electronic design and the number of technologies that must be mastered to bring market winning products have forced electronic companies to focus on their competence. Product specification, IP creation, design assembly and manufacturing are, for the most part, no longer taking place in the same organization. Indeed the electronic industry has been disaggregating from a vertically oriented model into a horizontally oriented one. Time-to-market pressure, design complexity and cost of ownership for mask are driving towards more disciplined design style that favor design re-use and correct-the-first time implementations. The quest for flexibility in embedded system design coupled with the previous considerations is pushing the electronic industry towards programmable solutions for a larger class of designs than ever before. Design methodology has become THE focus: design infrastructure and tools must be developed in synchrony with design methodology. The methodology is based on defining *Platforms* at the key articulation points in the design flow. Each platform represents a layer in the design flow for which underlying, subsequent design-flow steps are abstracted. By carefully defining the platform layers, developing new representations and associated transitions from one platform to the next. Various forms of platform-based design have been used for many years. My intention is to formally define the key principles of platform-based design following the idea of Sangiovanni Vincentelli [3, 4, 5] that serves as a framework for design technology research and practices. The platform-based design methodology is the outgrowth of the SoC debate where the economics of chip manufacturing and design has been carefully studied in [6]. The overall goal of electronic system design is to balance production costs with development time and cost in view of performance, functionality and product-volume constraints.

1. Manufacturing cost depends mainly on the hardware components of the product. Minimizing production cost is the result of a balance between competing criteria. The size of the chip is an important factor in determining production cost. Minimizing the chip size implies tailoring the hardware architecture.
2. NRE (Non-Recurrent Engineering) costs associated with the design and tooling of complex chips is growing rapidly. The ITRS predicts that while manufacturing complex SoC design will be feasible, at least down to 40nm minimum feature sizes. Further-

more, the cost of developing and implementing a comprehensive test for such complex design will continue to represent an increasing fraction of total design cost unless new approaches are developed.

3. Designs costs are exponentially rising due to the increased complexity of the products, the challenges posed by physical effects for deep sub-micron and the limited human resources. Design productivity is falling behind exponentially with respect to the technology advances. Time-to-market constraints are also growing at such pace that even if costs were not an issue, it is becoming plainly impossible to develop complex parts within the constraints. An additional problem is the lack of skilled work force that could implement future IC´s considering the system aspects of the design and all second order physical effects that will be of primary importance in deep sub micron. One of the objectives of this thesis is the design methodology (i.e. Platform Based Design) that can trade off various components of manufacturing, NRE and design costs sacrificing as little as possible "potential" design performance.

Following the definition methodology in [7] where it is applied to all levels of design abstraction thereby providing an all-encompassing intellectual framework in which research and design practices can be embedded and justified. The concept of platform has been around for years. However, many are the definition of platforms that have been used and that depend on the domain of application. In the IC domain a platform considered a "flexible" integrated circuit where customization for a particular application is achieved by "programming" one or more of the components of the chip. Programming may imply "metal customization" (Gate arrays), electrical modification (FPGA personalization) or software to run on a microprocessor or a DSP. For example, the DaVinci [8] platform for multimedia processing, the Nexperia platform [9], Altera Stratix's [10], and the Xilinx Virtex's platform [11] are a few examples of this approach.

Definition 1. *"Platform-based design is defined as the creation of a stable microprocessor-based architecture that can be rapidly extended, customized for a range of applications, and delivered to customers for quick deployment" Source: Jean-Marc Chateau (ST micro).*

The basic tenets of the platform-based design methodology proposed in [12] are:

• Regarding design as a "meeting-in-the middle process" where successive refinements of specifications meet with abstraction of potential implementations;

• The precisely defined layers identification; where the refinement and abstraction process take place. The layer then support designs built upon them isolating from lower-level details but letting enough information transpire about lower levels of abstraction to allow design space exploration with a fairly accurate prediction of the properties of the final implementation. The information should be incorporated in appropriate parameters that annotated design choices at the present layer of abstraction. These layers of abstractions are called Platforms to stress their role in the design and their solidity.

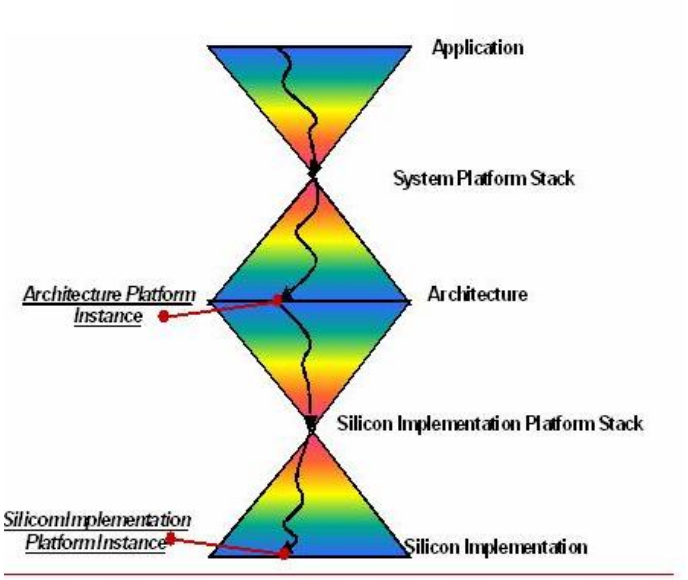


Figure 1.1: Platform Design Methodologies: Platform Stacks

Definition 2. ***Platform:*** *From a historical perspective and the newly concepts mentioned above, we can state that the general definition of a platform is an abstraction layer in the design flow that facilitate a number of possible refinements into a subsequent abstraction layer (platform) in the design flow [13].*

Definition 3.

Key to the application of the design principle is the careful definition of the platform layers. Platforms can be defined at several point of the design process. Some levels of abstraction are more important than others in the overall design trade-off space. In particular, the articulation point between system definition and implementation is a critical one for design quality and time. Indeed, the very notion of platform-based design originated at this point (see Ferrari and Sangiovanni-Vicentelli, 1999, Martin Chang et al, 1999, Balarin et al., 1997, Keutzer et al, 2000). According to Sangiovanni Vicentelli, there are two distinct kinds of platforms that will be mainly used in our work. Each one has their methods and tools necessary to link the two approximations. The one that has been studied the most is what they call the Architecture platform and the other one is the Network platform.

1.2 Architecture Platform and Network Platform

Integrated circuits used for embedded systems will most likely be developed as an instance of a particular architecture platform. That is, rather than being assembled from a collection of independently developed blocks of silicon functionality, they will be derived from a specific "family" of microarchitectures, possibly oriented towards a particular class of problems, that can be modified (extended or reduced) by the system developer. The elements of this family are a sort of "hardware" denominator that could be shared across multiple applications. Hence, the architecture platform concept as a family of architectures that are closely related is mainly geared towards optimizing design time: Every element of the family can be obtained quickly by personalizing an appropriate set of parameters that control the micro-architecture. This approach conjugates the need of saving design time with the optimization of the elements of the family for the application at hand. Architecture platforms are in general characterized by (but no limited to) the presence of programmable components so that each of the platforms instances that can be derived from the architecture platform maintain enough flexibility to support an application space that guarantees the production volumes required for economically viable manufacturing. The library that defines the architecture platform may also contain re-configurable components. Reconfiguration comes into two flavors: First, run-time reconfigurability such as Xilinx/Altera, where FPGA blocks can be customized by the user without the need of changing mask set, thus saving both design cost and fabrication cost. Design-time reconfigutability, such as Tensilica Xtensa [14], where the silicon is still application-specific in this case, only design time is reduced.

To implement communication networks, the functionality is mapped onto a Network Platform (NP) [15] that consist of a set of processing and storage elements (nodes) and physical media (channels) carrying the synchronization and data messages exchanged between nodes. Nodes and channels are the architecture resources in the NP library and are identified by parameters like processing power and storage size (nodes) and bandwidth, delay, error rate (channels). The task of choosing an NP requires selecting from NP library an appropriate set of resources and a network topology that defines how channels connect nodes. A broad range of options of physical channels (e.g. cable, wireless link, fiber) and network topologies (mesh, torus, fat tree, star, ring. etc.) are usually available. Therefore, the design space to explore is quite large.

1.3 Taxonomy of Architecture Platforms for Multimedia

In the early days of video coding technology, systems tended to fall into two categories, dedicated hardware designs for real-time video coding (e.g. H.261 videophones) or software designs for ´off-line´ (not real-time) image or video coding (e.g. JPEG, MJPEG or MPEG-2/4 compression/decompression software). The continued increases in processor performance, memory density and storage capacity have led to a blurring of these distinctions and video coding applications are now implemented on a wide range of processing platforms. General-purpose platforms such as personal computer (PC) processors can

achieve respectable real-time coding performance and benefit from economies of scale (i.e. widespread availability, good development tools, competitive cost). There is still a need for dedicated hardware architectures in certain niche applications, such as high end-end video encoding or very low power systems. The ´middle ground´ between the general-purpose platform and the dedicated hardware design (for design that requires more processing power than a general purpose processor can provide but where a dedicated design is not feasible) was, until recently, occupied by programmable ´video processors´. So-called ´media processors´, providing support for wider functionalities such as audio and communications processing, are beginning to occupy this market. There is currently a convergence of processing platforms, with media extensions and features being added to traditionally distinct processor families (embedded, DSP, general-purpose) so that the choice of platform for video CODEC design is wider than ever before. Platforms for Multimedia can be divided in platforms based in dedicated Hardware (custom platforms) and platforms based in Instruction Set processors as already mentioned in beginning chapter 1. The "programming" platforms Hardware are based in FPGAs or ASICs. The way to configure these platforms is with HDL (Hardware Description Languages). The Instruction Set Processor Platforms are based in devices to program the devices with languages soft such as: C, C++ or Java. In this part of the chapter we attempt to categorize the main platforms alternatives and to compare their advantages and disadvantages for the designer of a video coding system.

1.4 MPSoC Platform

1.4.1 Platform Definition

Multiprocessor systems-on-chip (MPSoC) are moving from simple multi-core structures with shared memory to many-core structures with distributed architecture [16]. Communication system of MPSoC can be based on bus interconnections, direct inter-node and indirect interconnections, network interconnections (using switches). Due to performance and power limitations buses may be used only in future MPSoC with number of nodes less than a dozen, [17, 18]. In many-core MPSoC a chip includes a set of computing nodes, processing elements (PE) with some memory, and direct interconnections between the nodes. Nodes and links between them form some mesh, a graph of nodes and links (it is common to call it "interconnection topology"). Nodes exchange data and control information using packets, messages, transactions,... Interconnection network determine many important characteristics of MPSoC as a whole. As the transistor size is scaling down, interconnectivity becomes more important in chip design. For instance, in 65 nm technology, delay caused by 1 mm interconnect is larger than the gate delay. Design moved from chip-wide synchronous circuitry to GALS (globally asynchronous, locally synchronous) many core chips design. Defining and optimization of interconnections for distributed architecture MPSoC is an important problem in MPSoC system level design. It is crucial to balance nodes and interconnection features, along with MPSoC properties, tasks requirements and VLSI technology features. Problem of power dissipation and power density is also very important for systems-on-chip in deep submicron technologies

[19]. The problem is that many features of on-chip wiring, its routing and placement cannot be defined at the system level stage of an MPSoC design. Nevertheless, system level design should include development and balancing a structure of a many-core MPSoC, its nodes and interconnection topology between them. General models and based on MPSoC methodology are required for system-level interconnection design in prospective MPSoC. Along with interconnection topology development in system level design, designers should select the way to implement its links (and switching nodes, if any) . Serial, parallel, multi-line links are used in current MPSoC implementations. Different standards require different chip resources for the implementation of links and nodes , having different maximal link length, different data rates, different power consumption and power density.

1.4.2 Communication within Process Elements (PE) tiles

There are two basic ways to connect several PE in a MPSoC, with buses or with network infrastructures.

A bus-based system has three types of modules: masters, slaves and arbitration. Master is active player in bus transactions, requesting services from other masters of from slaves in these transactions. Because a bus is a shared communication link, only one master at a time can access the bus, and that makes arbitration between masters necessary. The arbitration can be either centralized or distributed. A summarized explanation of communication within PE is made in the next paragraphs.

In the connexion with buses, there is a tendency to use exist segmented (and hierarchical) buses that are faster and use lower power per data transaction than their equivalent non-segmented ones. In segmented buses, different masters can start transactions in parallel. But, the revolution, driven by the technology capacity to build trillions of transistors in a single piece of silicon , let to the possibility of using hundreds of interconnected PEs .

1.4.3 Segmented Buses

The segmented bus structure enables faster operation than a conventional bus system and also offers lower power consumption per transferred data item. This is possible because segmentation is realized in such a way that the majority of data transfers in the system are intra segment on relative short wires with low or moderate capacitive loads [20].

Designing a segmented bus has two main aspects. First, the bus structure must be established, and this requires a thorough knowledge of the application that the devices will eventually serve. To compute the relative bus utilization factor for each master-slave pair, it is necessary to extract the relevant information about actual communication frequencies or probabilities between all masters and slaves in the system. This analysis makes possible to determine efficient system architectures. Certain algorithms [21, 22, 23] lead to an optimal solution as well as to the placement of system components on specific segments. The optimal configuration would provide a high communication probability

between components within the same segment limits. At the next level, placement of segments relative of each other should follow the frequency of transactions between their modules.

Second, the designer must establish a communication protocol and timing convention as a basis on which the participants in a transaction can proceed clearly and reliably . Synchronous bus architecture links the protocol events to a global bus clock. Asynchronous bus designs would rely on sequencing the events using self-timed handshaking signaling on asynchronous communication channels [24].

The segmented buses are a technique for communicating different IPs (microprocessor, coprocessor, ASIC, etc.) to execute diverse tasks. here is not an optimal synthesis and place and route (SPR) for every situation. Therefore, the current research in the field is investigating how to realize a correct task scheduling in the SoC. In this case, SoC understood as a set of IP with its inter-communication. If the segmented buses are optimal in the inter tile (IP) communication, then they are also energy efficient. Unfortunately, one of the problems of segmented buses is that they do not scale. If we have an implementation with a given set of IPs connected with segmented buses and we would like to connect some further IPs, we have to redo the study about its placement to optimize bandwidth while minimizing energy.

1.4.4 Network on Chip (NoC)

For solving the problem of scaling, researchers followed the point of view of a SoC as a micro network of components [25]. The network is the abstraction of the communication among components and must satisfy quality-of-service requirements —such as reliability, performance, and energy bounds— under the limitation of intrinsically unreliable signal transmission and significant communication delays on wires. SoCs differ from wide area networks in their local proximity and because they exhibit less non-determinism. Local, high-performance networks—such as those developed for large-scale multiprocessors— have similar requirements and constraints. However, some distinctive characteristics, such as energy constraints and design-time specialization, are unique to SoC networks . Whereas computation and storage energy greatly benefit from device scaling, which provides smaller gates and memory cells, the energy for global communication does not scale down. On the contrary, as the "Wiring Delays" sidebar indicates, projections based on current delay optimization techniques for global wires [26] show that global on-chip communication will require increasingly higher energy consumption. Hence, minimizing the energy used for communications will be a growing concern in future technologies. Further, network traffic control and monitoring can help to get a better power management than networked computational resources consume. For example, the clock speed and voltage of end nodes can vary according to available network bandwidth.

1.4.5 IP block or core

This paragraph explains what are Intellectual Property (IP) cores and their increasing

importance for SoC development. In electronic design, a semiconductor intellectual property core, IP block, or IP core is a reusable unit of logic, cell, or chip layout design and it is also the intellectual property of one party [27]. In digital-logic applications, IP cores are typically offered as generic gate netlists. A netlist is a boolean-algebra representation (gates, standard cells) of the IP's logical-function, analogous to an assembly-code listing for a high-level program application. Synthesisable cores are delivered in a hardware description language such as Verilog or VHDL, permitting customer modification (at the functional level). Both netlists and synthesisable cores are called soft cores', as both follow the SPR design-flow . Analog and mixed-signal logic generally require a lower-level, physical description. Hence, analog IP (SERDES, PLLs, DAC, ADC, etc.) are distributed in transistor-layout format (such as GDSII). Digital IP-cores are sometimes offered in layout format, as well. Such cores, whether analog or digital, are called 'hard cores' (or hard macros), because the core's application function cannot be meaningfully modified by the customer. Transistor layouts must obey the target foundry's process design rules, and hence, hard cores delivered for one foundry's process cannot be easily ported to a different process or foundry. For digital applications, soft cores and hard cores serve different roles. Soft-cores offer greater customer flexibility, while hard-cores, by the nature of their low-level representation, offer better predictability in terms of timing-performance and area. IP cores suppose a formalization oriented to design reuse, so that they may be licensed to another party or can also be owned and used by a single party alone (i.e. for different divisions of a large company). The term is derived from the licensing of the patent and source code copyright intellectual property rights that subsist in the design but also due to the business model associated to its behavior as a virtual component that can be placed in a piece of silicon like physical components were placed in PCBs. An uncommon alternative expansion is "integrated processor block". IP cores can be used as building blocks within ASIC chip designs or FPGA logic designs. IP cores in the electronic design industry have had a profound impact on the design of SoCs. The IP core can be described as being for chip design what a library is for computer programming or a discrete integrated circuit component is for printed circuit board design.

1.5 Custom and Configurable Processor Platform

The common tendency in processor development has been from multi-core to many-core: from dual-, quad-, eight-core chips to chips with tens or even hundreds of cores. Additionally, multi-core chips mixed with simultaneous multi-threading, memory-on-chip, and special-purpose "heterogeneous" cores promise further performance and efficiency gains, especially in processing multimedia, recognition and networking applications. Heterogeneous cores are though as a set of processors with different instruction sets, where each core is oriented to a dedicated application. Some examples of heterogeneous cores are listed in 1.5.1.There is also a trend of improving energy efficiency by focusing on performance-per-watt with advanced fine-grain or ultra fine-grain power management and dynamic voltage and frequency scaling (DVFS).

1.5.1 Mono Core and Multi Core: WLP, ILP and TLP

Software platforms can be divided in mono core and multi-core. At the same time, as manufacturing technology continues reducing the gate size, physical semiconductor-based microelectronics restrictions have become a key design concern. Some effects of these physical boundaries can be source of significant heat dissipation, device faults and data synchronization problems. The demand for more capable microprocessors causes CPU designers to use various methods of increasing performance. For example,Word Level Parallelism (**WLP**) or SIMD (Single Instruction, Multiple Data) is a technique employed to achieve data level parallelism. An application that may take advantage of SIMD is one where the same operation is being computed to a large number of data points. This common operation is quite common in many multimedia applications due that they are data dominant applications. Disadvantages of WLP are that not all algorithms can be vectorized. For example, a flow-control-heavy task like code parsing would not benefit from SIMD. Currently, implementing an algorithm with SIMD instructions usually requires human labor; for instance, most compilers do not generate SIMD instructions from a typical C program. Vectorization in compilers is an active area of computer science research. Another example, Instruction-Level Parallelism (**ILP**) methods like super-scalar pipelining are appropriate for many applications, but are inefficient for others that tend to contain difficult-to-predict code. A lot of applications are better suited to thread level parallelism (**TLP**) techniques, and multiple independent CPUs is one common method employed to increase a system's overall TLP. A combination of increased available space due to sophisticated manufacturing processes and the demand for improved TLP is the reason behind the conception of multi-core CPUs. Additionally, for general-purpose processors (GPP), much of the incentive for multi-core processors come up significantly due to diminished gains in processor performance from increasing the operating frequency (frequency-scaling). The memory wall and the ILP wall are the reasons why system performance has not increased as much as sustained processor frequency, as was previously observed. The memory wall refers to the growing gap between processor and memory speeds, driven by larger cache sizes to cover the latency to memory which assists to the extent that memory bandwidth is not the bottleneck in performance. The ILP wall refers to the growing complexity to find adequate parallelism in the instructions stream of a single process to maintain higher performance processor cores busy. Finally, the often cited, power wall refers to the trend of consuming double the power with each doubling of operating frequency (which is possible to contain to just doubling only if the processor is made smaller). The power wall causes manufacturing, system design and exploitation problems that have not been justified in the face of the diminished gains in performance due to the memory wall and ILP wall. Together, these three walls combine to motivate multi-core processors. A multi-core microprocessor (or chip-level multiprocessor, CMP) is one that combines two or more independent processors into a single package, often a single integrated circuit (IC). A dual-core device contains two independent microprocessors and a quad-core device contains four microprocessors. A multi-core microprocessor implements multiprocessing in a single physical package. Cores in a multi-core device may share a single coherent cache

at the highest on-device cache level (e.g. L2 for the Intel Core 2) or may have separate caches (e.g. current AMD quad-core processors [28]). The processors also share the same interconnect to the rest of the system. Each "core" independently implements optimizations such as super-scalar execution, pipelining, and multi-threading. A system with N cores is effective when it is presented with N or more threads concurrently. There has been a proliferation of multiprocessor computing platforms for scientific and embedded computing domains, as well as for the general purpose computing domain. Reference [29] looks at a set of recent multiprocessor platforms, and also provide some tools for analyzing a multiprocessor platform.

1.5.2 Advantages and disadvantages of Multi Core

Advantage

The immediacy of several CPU cores on the same die has the advantage that the cache coherence circuitry can operate at a much higher clock rate than if the signals go off-chip. Therefore, combining equivalent CPUs on a single die, considerably improves the performance of cache snoop operations. This means that, because signals between different CPUs travel shorter distances, those signals degrade less. These superior quality signals let more data to be sent in a given time phase since individual signals do not require to be repeated so frequently. Also, a dual-core processor uses slightly less power than two coupled single-core processors, principally because of the increased power required to drive signals external to the chip and because the smaller silicon process geometry allows cores to operate at lower voltages; such reduction reduces latency. Supposing that the die can fit into the package, physically, the multi-core CPU designs require much less Printed Circuit Board (PCB) space than multi-chip SMP designs. Furthermore, the cores share some circuitry, such as the L2 cache and interface to the front side bus (FSB). In conditions of competing technologies for the available silicon dies area, multi-core design can make use of proven CPU core library designs and produce a product with lower risk of design error than planning an original broader core design. As well, by adding more cache, returns diminish.

Disadvantages

Integration of a multi-core chip drives production yields down and they are more difficult to manage thermally than lower-density single-chip designs. From an architectural point of view, ultimately, single CPU designs may make better use of the silicon surface area than multiprocessing cores, so a development commitment to this architecture may carry the risk of obsolescence. In addition to operating system (OS) support, adjustments to existing software are required to maximize utilization of the computing resources provided by multi-core processors. Also, the ability of multi-core processors to increase application performance depends on the use of multiple threads within applications. For example, most current (November 2008) [2] explains that for some computing tasks, 8 cores are not (yet) much better than 4 cores.

1.5.3 Homogeneous and Heterogeneous Processor Platform

The multi-core software platforms can be divided in homogeneous platforms and heterogeneous platforms. We define an homogeneous processor as a processor that is based in one or several ISP cores processor, all with similar features. Then, the heterogeneous Processor platform is based in different cores with specialized task each one. Typical examples of homogeneous processors are:

Homogeneous Examples (AMD and INTEL): AMD family K10 and K10.5 processors Phenon 65nm SOI triple-core and PhenonII with 45nm SOI process QuadCore[30]. Intel will launch its 6 cores (12 threads with multi-threading) processor named Westmere (formerly Nehalem-C) is the name given to the 32 nm shrink of Nehalem. Westmere should be ready for a 2009 release provided that Intel stays on target with its road map. However, it appears that the bulk of Westmere's versions, excluding mobile versions, will be released sometime in 2010 [31, 32, 33].

Heterogeneous Examples (ARM, DaVinci and Cell Processor): ARM MPCore is a fully synthesisable multi-core container for ARM9 and ARM11 processor cores, intended for high-performance embedded and entertainment applications. A new type of SoC processor has appeared that integrates high-performance and programmable cores, together with the essential memory and peripherals for building a wide range of consumer video systems. The typical SoC architecture for heterogeneous processors based on a programmable digital signal processor (DSP) with video-specific hardware acceleration providing the computational performance needed for real-time compression-decompression algorithms (codecs) and other communications signal processing.

Combining a RISC processor with the DSP lets to platforms with control and user interface support, (including programming and operating system facilities) and integrated video peripherals, that reduce system cost and simplify design. Since this multiprocessor hardware serves as the foundation for open software architecture, the result is an SoC processing engine that enables the flexible, rapid development of robust consumer video products. Three examples of heterogeneous processors related with video processing are: (1)MSC8156E [34] is a MSC8156 processor with a six-core DSP at 1GHz based on Freescale's new SC3850 StarCore technology and designed to advance the capabilities of wireless broadband equipment. It delivers performance and power savings, leveraging 45 nm process technology; (2) DaVinci/OMAP[35] a DSP processor with an embedded ARM processor from TI; (3)IBM's Cell processor [36, 37] can be an example of one GPP with several media processors.

1.6 Architectural Options

There are different Architectural options for hardware design implementation such as: General purpose processor, we can provide as example of this architecture a Reduced Instruction Set Computer (RISC) or a Complex Instruction Set Computing (CISC)

processors. Another possibility is a Domain Specific Processor such as an Application Specific Instruction Processor (ASIP). An additional possibility is a Custom Processor Architecture such as an Application Specific Integrated Circuit (ASIC) and finally, we can consider as an option of our design a Reconfigurable Architecture such as a FPGA or CPLD. In the following sections 1.7.1 to 1.8, there is a taxonomy and each solution is explained more accurately.

1.6.1 General-Purpose Processor Platform

A desktop PC contains a processor (with mono or multiple homogeneous cores) that can be described as general-purpose. The processor is designed to provide acceptable performance for a wide range of applications such as office, games and communications applications. Manufacturers need to balance users demand for higher performance against the need to keep reduced costs for a mass-market product. At the same time, the large economies of scale in the PC market make possible, for the major manufacturers, to rapidly develop and release higher-performance versions of the processors. Table 1.1 list some of the main players in the market and their recent processors offering (as of January 2009):

Manufacturer	Last offering	Features
Intel eight (16) cores	Beckton	45nm Hi-k TDP: 130/105/90W,Chipset 4xQPI, Cache 24MiB release Q3-Q4 2009 [38]
Intel six (12)	Westmere	32nm, Q4-2009 to Q2-2010 [39]
IBM (2 chips per module)	Power7	45 nm , 4.0GHz, 256 GFLOPS per chip [40]
(4 threads per core (32 per chip))		(more than 781M transistors)
AMD Deneb (4 quad-core)	Phenon II X4 950	45 nm TDP 125W, IMC 3.1 GHz
		L2 4×512 KB L3 6 MB rel.Q2 2009 [41]

Table 1.1: Popular PC Processors

For more details on this topic, see List of future Intel [42] and AMD Phenom microprocessors [43]

Capabilities PC processors can be loosely characterized as good performance when running general applications; but they are not optimized for any particular class of application (though the trend is to add features such as SIMD capabilities to support multimedia applications). One of their main drawbacks is their high power consumption (though some lower-power versions of the above processors are available for mobile devices). The popular PC operating systems (Windows, Mac O/S and GNU-Linux) support multi-tasking applications and offer good support for external hardware (via plug-in cards or interfaces such as USB, USB2, fire wire, PCI, or PCMCIA).

Advantages	Disadvantages
High market penetration, very large potential user base	Computationally intensive video coding functions must be carried out in software
Availability of efficient compilers and powerful development tools	Medium to high power consumption
Multimedia extensions to improve video processing performance	Use of ´special´ instructions such as SIMD limits the portability of the video coding applications
Efficient multi-tasking with other applications	Processor resources not always available (can be problematic for real-time video)
Availability of multimedia application development framework	

Table 1.2: Advantages and Disadvantages of PC platform

Multimedia support Trends towards multimedia applications have let to increase support for real-time media. There are several frameworks that may be used between the Windows O/S, for example to assist in the rapid development of real-time applications. The DirectX and Windows Media framework provide standardized interfaces and tools to support efficient capture, processing, streaming and display of video and audio.

The increasing usage of multimedia has driven processors manufacturers to add architectural and instruction support for typical multimedia processing operations. The three processors families listed in table 1.1 (Beckton/Westmere, Power7 and PhenonII) support a version of single instruction multiple data (SIMD) processing. Intel's MMX and SIMD extensions (following Flynn taxonomy [44]) more info in chapter 5 that provides a number of instructions aimed at media processing. A SIMD instruction operates on multiple data elements simultaneously (e.g. multiple 16 bit words within a 64 bit or 128 bit register). This facilitates computationally intensive, repetitive operations such as motion estimation (e.g. calculating sum of differences, SAD) and DCT (e.g. multiple-accumulate operations). Intel later addressed these shortcomings with SSE, an expanded set of SIMD instructions with 32-bit floating point support and an additional set of 128-bit vector registers that made it easy to perform SIMD and FPU operations at the same time. SSE was in turn expanded with SSE2, which also extended MMX instructions so they can operate on 128-bit MMX registers. Support for any of these later instruction sets implies support for MMX. Intel's competitor AMD enhanced Intel's MMX with the 3DNow! instruction set. SSE2, Streaming "Single Instruction, Multiple Data" Extensions 2, is one of the IA-32 SIMD instruction sets. SSE2 was first introduced by Intel with the initial version of the Pentium 4 in 2001. It extends the earlier SSE instruction set, and is intended to fully supplant MMX. Intel extended SSE2 to create SSE3 in 2004. SSE2 added 144 new instructions to SSE, which has 70 instructions. Rival chip-maker

AMD added support for SSE2 with the introduction of their Opteron and Athlon 64 ranges of AMD64 64-bit CPUs in 2003.

Supplemental Streaming SIMD Extension 3 (SSSE3) is Intel's name for the SSE instruction set's fourth iteration. The previous state of the art was SSE3, and Intel have added an S rather than increment the version number, as they appear to consider it merely a revision of SSE3. Introduced in Intel's Core Microarchitecture, SSSE3 is available in the Xeon 5100 series (Server and Workstation) processors and the Intel Core 2 (Notebook and Desktop) processors. SSSE3 contains 16 new discrete instructions over SSE3. Each can act on 64-bit MMX or 128-bit MMX registers. Therefore, Intel's materials refer to 32 new instructions. The earlier SIMD instruction sets on the x86 platform, from oldest to newest, are MMX, 3DNow! (developed by AMD), SSE, 3DNow! Professional, SSE2, and SSE3.

SSE4 is a new instruction set for the Intel Core microarchitecture, initially implemented in the Penryn processor [45]. The SSE4 Programming Reference is available from Intel.

SSE4 consists of 54 instructions. A subset consisting of 47 instructions, referred to as SSE4.1 in some Intel documentation, will be available in Penryn. SSE4.1 and SSE4.2, a subset consisting of 7 instructions, will first be available in Nehalem (old Beckton).

TABLE 1.6.1 summarizes the main advantages and disadvantages of PC platforms for video coding applications. The large user base and comprehensive development support make it an attractive platform for applications such as desktop video conferencing .

1.6.2 Generic Digital Signal Processors

Digital Signal Processors (DSPs) are designed to efficiently handle applications that are based around computationally intensive signal processing algorithms. Typical applications include audio processing (e.g. filtering and compression), telecommunications functions (such as modem processing, filtering and echo cancellation) and signal conditioning (transformation, noise reduction, etc.). Mass-market applications for DSPs include PC modems, wireless and hand-held communications processing, speech coding and image processing. These applications typically require good signal processing performance in a power, cost and/or space-limited environments. DSPs can be characterized as high performance for a selected range of signal processing operations; low/medium power consumption; low/medium cost with fixed or point arithmetic capability; limited on- and off-chip code and data storage (depending on the available address space); 16- or 32- bit word length. Table 1.6.2 lists a few popular DSPs and compares their features: this is only a small selection of the wide range of DSPs on the market. As well as these discrete ICs, a number of manufacturers provide DSP cores (hardware architectures designed to be integrated with other modules on a single IC).

A key feature of DSPs is the ability to efficiently carry out repetitive processing algorithms such as filtering and transformation. This means that they are well suited to many of the computationally intensive functions required of a typical DCT-based video CODEC, such as motion estimation, DCT and quantization, and some promising perfor-

Licensor	Family	{2}	{3}	{4}	{5}	{6}
ARC	ARC 600/ARC XY	P	16/32	180	4 G	0.77
	ARC 700/ARC XY	P	16/32	265	4 G	1.1
ARM	ARM7	P	32	145	4 G	0.28
	ARM9	P	32	265	4 G	n/a
	ARM9E	P	16/32	265	4 G	1.7
	ARM1136	P	16/32	330	4 G	2.3
CEVA	CEVA-TeakLite	P	16	170	256 K	0.4
	CEVA-TeakLite II	P	16	200	4 M	0.5
	CEVA-Teak	P	16	150	8 M	0.9
	CEVA-X1620	P	8/16	330	4 G	2.6
MIPS	MIPS32 24KE	P	32	335	4 G	2.0
	(with DSP ASE)					
Philips	CoolFlux DSP	P	24	175	640 K	0.34
Tensilica	Diamond 545CK	P	18	230	4 G	3.7
VeriSilicon	ZSPneo	P	16/32	165	256 K	0.45
	ZSP200	P	16/32	165	256 K	0.7
	ZSP400	P	16/32	165	256 K	1.3
	ZSP410	P	16/32	185	4 G	1.4
	ZSP500	P	16/32	205	64 M	2.2
	ZSP540	P	16/32	200	64 M	2.7
	ZSP600	P	16/32	175	64 M	3.1

Table 1.3: Common DSP Cores ({0}Licensor::ARC, ARM,C:CEVA,M:MIPS,P:Philips, T:Tensilica, V:Verisilicon. {1} Family. {2} Floating, Fixed or Both, F,P,B.{3} Data Width in bits. {4} Core Clock Speed in MHz. {5}Total Core Memory Space, Bytes {6} die area in mm^2.)

NOTES

The ARC ARC 600/ARC XY family has Customizable core with optional DSP features while the ARC 700/ARC XY family has longer pipeline enables higher clock rate. The ARM7 is widely licensed 32-bit microprocessor core. The ARM9 adds separate bus for data access, deeper pipeline to ARM7. The ARM9E enhanced with single-cycle MAC unit. The ARM1136 adds SIMD, load/store unit, branch prediction, deeper pipeline. The CEVA licensor has the family CEVA-TeakLite and CEVA-TeakLite II that is faster than previous one. The CEVA-Teak has dual-MAC DSP core and the CEVA-X1620 has 8-way VLIW, dual-MAC DSP core. The MIPS licensor has the MIPS32 24KE family that is a MIPS core with SIMD DSP extensions. The Philips licensor has the family CoolFlux DSP that has a Dual-MAC core targets low-power audio applications. Tensilica has the family Diamond 545CKthat is a VLIW-based customizable core; with optional DSP features. And finally, VeriSilicon has ZSPneo family that is a single-MAC, scalar variant of the ZSP400. ZSP200 that is single-MAC, 2-way superscalar variant of the ZSP400. ZSP400 that is Dual-MAC, 4-way superscalar DSP core. ZSP410 that is an enhanced ZSP400 version with instruction cache. ZSP500 that is second-generation ZSP; dual-MAC, 4-way superscalar. ZSP540 that is a Quad-MAC, 4-way variant of the ZSP500 and finally the ZSP600 that is a Quad-MAC, 6-way variant of the ZSP500.

mance results have been reported in [46, 47, 48, 49]. Because DSP is specially designed for this type of applications, this performance usually comes without the penalty of high power consumption. Support for related video processing functions (such as video capture, transmission and rendering) is likely to be limited. The choice of application development tools is not as wide as for the PC platform and high level support is often limited to the C language. Table 1.4 summarizes the advantages and disadvantages of the DSP platform for video coding applications.

In a typical DSP development scenario, code is developed on a host PC in C, cross-compiled and downloaded to a development board for testing. The development board consist of a DSP IC together with peripherals such as memory, A/D converters and other interfaces. To summarize, a DSP platform can provide good performance with low power consumption but operating system and development support is often limited. DSPs may be suitable platforms for low-power, special-purpose applications (e.g. a hand-held videophone). An example of state-of-the art DSP is DaVinci[35] from Texas Instruments.

Advantages	Disadvantages
Low power consumption	Less suited to higher-level complex
	aspects of processing
Relatively high computationally performance	Limited development support
Low price	Limited operating system support
Built-in telecommunications support (e.g. modem	Limited support for external devices
functions, A/D conversion)	(e.g. frame capture and display)

Table 1.4: Advantages and disadvantages of DSP platform

It is important to take into account that there is no easy way to evaluate DSP processor performance meaningfully [50]. Traditional performance units like MIPS and MOPS are misleading and do not reflect many relevant factors like application execution time, memory usage, and energy consumption. Application benchmarks, too, suffer from limitations that make fair comparisons difficult. More info about this in chapter 3 where in 3.2 the state of the art explains benchmarking. Fortunately, a methodology of algorithm kernel benchmarking and application profiling provides good estimates of processor performance weighted to the target application. We expect that DSP systems will continue to become more sophisticated and demand greater computational performance. At the same time, semiconductor vendors will continue to develop more powerful DSP processors and integrate these processors with other system components such as microcontrollers and peripherals. As systems become more complicated and processor choices grow, designers will need good estimates of a processor's DSP performance.

1.6.3 Media Processors

DSPs have certain advantages over GPPs for video coding applications; so-called media-processors go to a step further by providing dedicated hardware functions that support video and audio compression and processing. The general concept of a media processor is a core processor together with a number of dedicated co-processor that carry out application-specific functions.

The core of a Media Processor architecture is a very long instruction word (VLIW) processor. A VLIW processor can carry out operations on multiple data words (typically four 32 bit words in the case of a TriMedia processor [51, 52] at the same time. This is a similar concept to the SIMD instructions described chapter 5 and it is useful for video and audio coding applications. Computationally intensive functions in a video CODEC such as motion estimation and DCT may be efficiently carry out using VLIW instructions. Some advantages and disadvantages can be viewed in the table 1.5.

Advantages	Disadvantages
Low power consumption	Limited performance
Low price	Limited word lengths, arithmetic, address spaces
High market penetration	(As yet) few features to support video processing
Good development tool support	
Increasing performance	

Table 1.5: Advantages and disadvantages of media processors

1.6.4 Video Signal Processors

Video signal processors are positioned between media processors (with aim to process multiple media efficiently) and dedicated hardware CODECs (designed to deal with one video coding standard or a limited range of standards). A video signal processor contains dedicated units for carrying out common video coding functions (such as motion estimation, DCT/IDCT and VLE/VLD) but allows a certain degree of programmability, enabling a common platform to support a number of standards and to be at least partly future proof (i.e. capable of supporting future extensions new standards). Table 1.6 summarizes the advantages and disadvantages of this type of processor. Video signal processors do not appear to be a strong force in the video communications market, perhaps because they can be outperformed by a dedicated hardware design while they do not offer the same flexibility as a media processor or general-purpose processor.

Advantages	Disadvantages
Good performance for video coding	Application-specific features
	may not support future coding standards
Application-specific features	(but generally more flexible than dedicated HW)
Limited programmability	Reprogramming likely to require high effort
	Limited development tool support
	Cost tend to be relatively high for mass market
	applications
	Dependent on a single manufacturer

Table 1.6: Advantages and Disadvantages of video signal processors

1.6.5 Co-Processors

A co-processor is a separate unit that is designed to work with a host processor (such as a general-purpose PC processor). The co-processor (or accelerator) carries out certain computationally intensive functions in hardware, removing some of the burden from the host. Complex, standard-specific functions such as variable length decoder and header parsing are carried out by the host while computationally intensive functions (but relatively regular and common to most standards) such as IDCT and motion compensation are offloaded to the accelerator. The basic operations of this type of systems are as follows:

1 . The host decodes the bit stream and extracts rescaled block coefficients, motion vectors and header information.

2 . This information is passed to the accelerator (using a standard API) via a set of data buffers.

3 . The accelerator carries out IDCT and motion compensation and writes the reconstructed frame to a display buffer.

4 . The display buffer is displayed on the PC screen and is also used as a prediction for further reconstructed frames.

Advantages	Disadvantages
High performance for video coding	No support for other coding standards
Optimized for target video coding standard	Limited control options
Cost-effective for mass market applications	Dependent on a single manufacturer

Table 1.7: Advantages and disadvantages of co-processor architecture

Table 1.7 lists the advantages and disadvantages of this type of system. The flexibility of software programmability, together with dedicated hardware support for key functions, makes it an attractive option PC-based video applications. Developers should benefit from the large PC market which will tend to ensure competitive pricing and performance for the technology.

Example of GPP with a Coprocessor (MPC8220i: PowerPC-Based ASSP Designed For Imaging Applications)

One example of GPP and coprocessor specialized in imaging task is the Freescale's MPC8220(i) [53] system-on-chip (SoC) is a versatile integrated microprocessor (a superscalar PowerPC architecture delivering 567 MIPS (Dhrystone) at 300MHz) with peripheral combinations that can be used in a variety of consumer electronic products. It embeds an enhanced version of the high-performance PowerPC (PPC) 32-bit 603e core processor with integrated features such as fast Ethernet controller (FEC), universal serial bus (USB), single/double data rate (SDR/DDR), synchronous dynamic random access memory (SDRAM), and peripheral component interconnect (PCI) bus. The MPC8220 family builds on the legacy of instruction set architecture (ISA)-compatible devices from the MPC5xxx and MPC822x lines. The MPC8220i includes unique image coprocessor technology that enhances performance on data-intensive graphics applications. This combination of features makes the MPC8220 family well suited to consumer products for imaging and entertainment, network appliances, Internet-access devices, and wireless access control. A GNU/Linux version has been released for the MPC8220(i), enabling designers to experience a complete GNU/Linux solution for its MPC8220-based platform.

1.7 Domain Specific Integrated Processors

An application-specific instruction-set processor (ASIP) [54, 55, 56] is a component used in System-on-a-Chip design. The instruction set of an ASIP is tailored to benefit a specific application. This specialization of the core provides a trade off between the flexibility of a general purpose CPU and the performance of an ASIC. Some ASIPs have a configurable instruction set. Usually, these cores are divided into two parts: static logic which defines a minimum ISA and configurable logic which can be used to design new instructions. The configurable logic can be programmed either in the field similarly

to an FPGA or during the chip synthesis.

1.7.1 Instruction Set Processor Platforms

These devices can be based in RISC processors or CISC processors. RISC, The Reduced Instruction Set Computer, is a CPU design philosophy that favors a reduced instruction set as well as a simpler set of instructions. The idea was originally inspired by the discovery that many of the features that were included in traditional CPU designs to facilitate coding were being ignored by the programs that were running on them. Also these more complex features took several processor cycles to be performed. Additionally, the performance gap between the processor and main memory is increasing. This leads a number of techniques to streamline processing within the CPU, while at the same time attempting to reduce the total number of memory accesses. The most common RISC microprocessors are Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, Power Architecture, and SPARC. A Complex Instruction Set Computer (CISC) is a microprocessor in which each instruction can execute several low-level operations, such as a load from memory, an arithmetic operation, and a memory store, all in a single instruction. The term was retroactively coined in contrast to reduced instruction set computer (RISC). The first highly pipelined "CISC" implementations, such as 486s from Intel, AMD, Cyrix, and IBM, certainly supported every instruction that their predecessors did, but achieved high efficiency only on a fairly simple x86 subset (resembling a RISC instruction set, but without the load-store limitations of RISC). Modern x86 processors also decode and split more complex instructions into a series of smaller internal "micro-operations" which can thereby be executed in a pipelined (parallel) fashion, thus achieving high performance on a much larger subset of instructions. Very Long Instruction Word or VLIW refers to a CPU architectural approach in order to take advantage of instruction level parallelism (ILP). A processor that executes every instruction one after the other (i.e. a non-pipelined scalar architecture) may use processor resources inefficiently, potentially leading to poor performance. The performance can be improved by executing different sub-steps of sequential instructions simultaneously (this is pipelining), or even executing multiple instructions entirely simultaneously as in superscalar architectures. Further improvement can be realized by executing instructions in a different order from the one they appear in the program; this is called out-of-order execution. These three techniques all have a cost: increased hardware complexity. Before executing any operations in parallel, the processor must verify that the instructions do not have interdependencies. There are many types of interdependencies, but a simple example would be a program in which the first instruction's result is used as an input for the second instruction. They clearly cannot be executed at the same time, and the second instruction can not be executed before the first one. Modern out-of-order processors use significant resources in order to take advantage of these techniques, since the scheduling of instructions must be determined dynamically. The VLIW approach, on the other hand, executes operations in parallel based on a fixed schedule determined when programs are compiled. Since determining the order of execution of operations (including the operations that can be executed simultaneously) is handled by the compiler, the processor does not need the

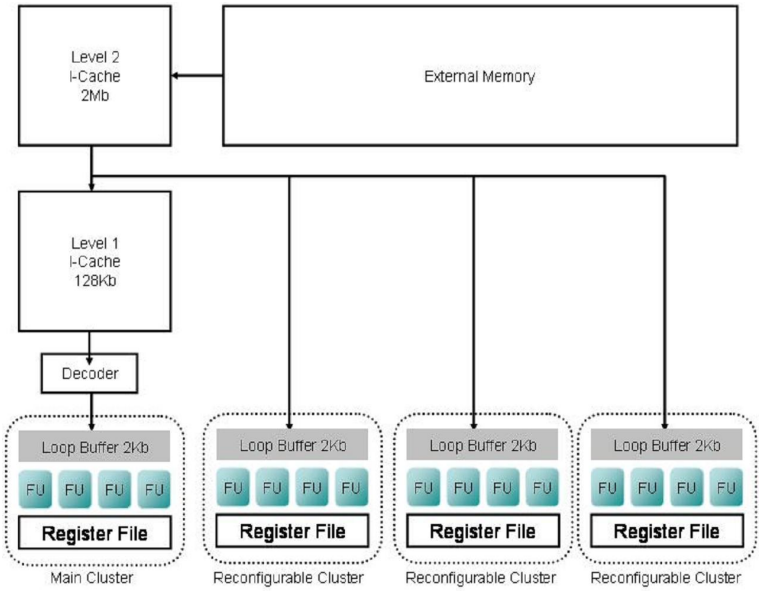


Figure 1.2: Example of CRISP instance (FU: Functional Unit)

scheduling hardware that the three techniques described above require. As a result, VLIW CPUs offer significant computational power with less hardware complexity (but greater compiler complexity) than those associated with most superscalar CPUs.

Example of Specific Integrated Processors (CRISP-Trimaran)

The CRISP framework [57] is a retargetable compiler and simulator framework based on Trimaran [58] (Trimedia Technologies Inc 1999). The architecture described by CRISP is VLIW based and, as all VLIW processors, the processor executes instructions that are composed of parallel operations. These operations are executed in the Functional Units (FUs) attached coarse-grained reconfigurable logic.

The processor's data path is divided in clusters (or slices), which contain one or more register files, one or more functional units, and bypass logic (aka. interconnect). An schema is shown in figure1.2. This well known division of the data path in clusters reduces the energy consumption and the delay of the data path by reducing the complexity of the register files and the bypass logic. Since reducing the number of cycles is also an effective way to reduce the energy consumption, all data path operations can be predicated to enable efficient execution of loops with conditionals. Furthermore, some chains of data dependent operations can be executed in a single clock cycle (in a fashion similar to FPGAs), using software controlled functional unit chaining, to decrease the effective latency of a set of operations, and hence reduce the number of clock cycles. Thanks to this chaining, some register file accesses can be prevented, further reducing the energy consumption. Compiler research optimizations doable are high-level machine independent code transformations and "back-end" machine dependent optimizations such as

Advantages	Disadvantages
Low power consumption	High development time
High performance	Limited development support
Complete word lengths, arithmetic, address spaces	Limited operating system support
features to support video processing	

Table 1.8: Advantages and disadvantages of ASIPs

instruction scheduling, register allocation, etc. A table 1.8 shows ASIP pros and cons.

Software Impact

Software benefits from multi core architectures where codes can be executed in parallel. Under most common operating systems this requires code to execute in separate threads or processes. Each application running on a system runs in its own process, so multiple applications will benefit from multi core architectures. Every application may also have multiple threads but, in most cases, it must be specially written to use many threads. Operating system software also tends to run many threads as a part of its normal operation. Running virtual machines will benefit from adoption of multiple core architectures since each virtual machine runs independently of others and can be executed in parallel.

Most application software is not written to intensively use multiple concurrent threads because of the challenge of doing so. A frequent pattern in multi-threaded application design is where a single thread does the intensive work while other threads do much less. Multi threaded code often requires complex co-ordination of threads and can easily introduce, subtle and difficult to find, bugs due to the interleaving of processing on data shared between threads (thread-safety). Debugging such code, when broken, is also much more difficult than single-threaded code. Also there has been a perceived lack of motivation for writing consumer-level threaded applications because of the relative rarity of consumer-level multiprocessor hardware. Although threaded applications incur little additional performance penalty on single-processor machines, the extra overhead of development was difficult to justify due to the preponderance of single-processor machines. Parallel programming techniques can benefit from multiple cores directly. Some existing parallel programming models such as OpenMP [59] and MPI [60, 61] can be used on multi-core platforms. Intel introduced a new abstraction for C++ parallelism called TBB (Threading Building Blocks).

Given the increasing emphasis on multi core chip design, branching from the grave thermal and power consumption problems posed by any further significant increase in processor clock speeds, the extent to which software can be multi threaded to take advantage of these new chips is likely to be the single greatest constraint on computer performance in the future. If developers are unable to design software to fully exploit the resources provided by multiple cores, then they will ultimately reach an insurmountable performance ceiling. In the following sections different ISP possible architecture are

presented.

1.8 Custom Hardware Architectures

Custom hardware for video coding uses specialized micro-architectures for computing parallel and concurrent operations based on the direct implementation of elementary functions from video compression and decompression algorithms. For example, building a specialized ASIC to accomplish some QoS (frame rate, quality) for a high end video management solution. Another case is to achieve a prototype in a FPGA platform. These architectures allow managing dedicated hardware operations, specialized memory and register clusters for system design. That permits a custom solution and therefore with higher performance but, in contrast, reduced flexibility.

1.8.1 Application Specific Integrated Circuit - ASIC platform

Application-Specific Integrated Circuits (ASIC) are integrated circuits customized for a particular use rather than general-purpose use. These devices improve performance compared to general-purpose CPUs, since ASICs are "hardwired" to perform a specific task and do not incur the overhead of fetching and interpreting stored instructions. However, a standard cell's ASIC may include one or more microprocessor cores and embedded software, in which case, it may be referred to as a "System on a Chip" (SoC). A full custom ASIC chip is the most costly, and like standard cell ASIC, uses a custom-designed mask for every layer in the chip. Unlike standard cells, designers of a full custom device have total control over the size and topology of every transistor forming every logic gate and regular structure, so they can "fine" tune each case to obtain the optimum performance. Thus, a full custom ASIC performs electronic operations as fast as possible, provided that the circuit design is efficiently architected.

In our design, we built an standard cell ASIC for the MPEG kernel part in 90nm UMC Faraday technology (Faraday refers to the cell libraries used for the UMC technology). We performed the placement and routing of the chip using Cadence tools. Further verification must be carried out around the back-end process, such as pre- and post-layout simulations, and test-vectors generated to ensure enough fault coverage (around 95% of nodes verified with the test vectors) concerning fabrication tests.

1.8.2 Field Programmable Gate Array - FPGA platform

Field-Programmable Gate Array (FPGA) is a semiconductor device containing programmable logic components called "logic blocks", programmable interconnects , I/Os and memory blocks where the logic can be programmed after it is manufactured. A hierarchy of programmable interconnects allows logic blocks to be organized as required by the system designer. Some FPGAs also have dedicated multipliers in specialized blocks like MAC (Multiply and Accumulate) blocks, different memory structures (multi-function, multi-ported, different sizes) and specific high-speed interconnect wiring. With

the increase in transistor density, modern FPGAs allow complete SoC systems to be instantiated with several processors. These embedded processors can be hard-coded in the FPGA, like the PowerPC and ARM processors [11, 62], or can be implemented using logic synthesis, like MicroBlaze for Xilinx and NIOS II [63] for Altera FPGAs.

General-purpose processor and media and video signal processors sacrifice a certain amount of performance in order to retain flexibility and programmability. A dedicated hardware design, optimized for a specific coding standard, is likely to offer the highest performance (in terms of video processing capacity and power consumption) at the expenses of rigidity. Some advantages and drawbacks are in table 1.9.

Advantages	Disadvantages
High performance for video coding	Limited control options
Optimized for target video coding standard	No support for other coding standards
Cost-effective for mass market applications	Dependent on a single manufacturer

Table 1.9: Advantages and Advantages and Disadvantages of dedicated hardware CODECs

Figure 1.3 attempts to compare the merits of the processing platforms discussed in this part. It should be emphasized that the rankings in this table are not exact and there will be exceptions in a number of cases (for example, a high-performance DSP that consumes more power than a media processor). However the general trend is probably correct: the best coding performance per milliwatt of consumed power should be achievable with a dedicated hardware design, but on the other hand PC and embedded platforms are likely to offer the maximum flexibility and the best development support due to their widespread usage.

Embedded processors

The term embedded processor usually refers to a processor or controller that is embedded into a larger system, in order to provide programmable control (usually non user programmable) and perhaps processing capabilities along side more specialized, dedicated devices. Embedded processors are widely used in communications (mobile devices, network devices, etc) and control applications (e.g. automotive control). Typical characteristics are low power consumption with restricted processing capabilities and addressing modes, limited word length and (sometimes) fixed-point arithmetic. In general, "embedded system" is not somehow fuzzy term, as many systems have some element of programmability. For example, handheld computers share some elements with embedded systems — such as the operating systems and microprocessors which power them — but are not truly embedded systems, because they allow user programming, different applications can be loaded and peripherals can be connected to them. Solutions presented in this work can be seen as embedded since all solutions are closed to the user,

who only can select (manually or automatically) balances between performance and low power. Sacrificing performance against a lower power consumption for systems where battery life can be considered as one of the main issues.

Therefore, embedded computing is a distinct area than "mainstream" PCs concerning processor technology. The same technological drivers towards multicore apply here too. Indeed, in many cases the application is a "natural" fit for multicore technologies if the task can easily be partitioned between the different processors.

Advantages	Disadvantages
Low power consumption	Limited performance
Low price	Limited word lengths, arithmetic, address
High market penetration	spaces
Good development tool support	(As yet) few features to support
Increasing performance	video processing

Table 1.10: Advantages and disadvantages of embedded processors

Until recently, an embedded processor would not have been considered suitable for video coding applications because of severely limited processing capabilities. However, as for other processor types , processing power of new generations of embedded processors continues to increase. Table 1.10 summarizes the features of some popular embedded processors.

The popular ARM and MIPS processors are licensed as cores for integration into third-party systems. ARM is actively targeting low-power video coding applications, demonstrating 15 frames per second H.264 encoding and decoding (QCIF resolution) on an ARM9 [64] and developing co-processor hardware to further improve video coding performance.

Manufacturer	Device	Features
MIPs	4K series	Low Power, 32 bit, up to approx 400MIPS
		multiply-accumulate support (4KM)
ARM	ARM9 series	Low power, 16bit up to 220MIPS
ARM/Intel	StrongARM series	Low power, 32bit up to 270MIPS

Table 1.11: Embedded processors features

Table 1.11 summarizes the advantages and disadvantages of embedded processors for video coding applications. Embedded processors are of interest because of their large market penetration (for example, in the high-volume mobile telephone market). Running low-complexity video coding functions in software on an embedded processor (perhaps

with limited hardware assistance) may be a cost-effective way of bringing video applications to mobile and wireless platforms. For example, the hand-held video-phone is seen as a key application for the emerging 3G high-rate mobile networks. Video coding on-low power embedded or DSP processor may be a key enabling technology for this type of device.

The recent trend is for a convergence between "so-called" dedicated media processors and general-purpose processors, for example demonstrated by the development of SIMD/VLIW-type functions for all the major PC processors. This trend is likely to continue as multimedia applications and services become increasingly important. At the same time, the latest generation of coding standards (e.g. MPEG-4 and H.264) require relatively complex processing (e.g. to support object-based coding and coding mode decisions), as well as repetitive signal processing functions such as block-based motion estimation and transform coding. These higher-level complexities are easier to handle in software than in dedicated hardware, and it may be that dedicated hardware CODECs will become less important (except for specialized, high-end such as studio encoding) and that general-purpose processors or co-processors to handle low-level signal processing).

1.9 Introduction to Languages for Architecture Description

Every design tool and model is used for solving a specific problem, but there is no tool for all purposes. A language that modelled everything from functional requirements to transistor layout would be too complicated, too large, and too complex to implement, maintain and train. Furthermore, in other to obtain meaningful implementation data it is not necessary to go below RTL. From Verilog or VHDL descriptions direct synthesis is already solved using EDA tools.

If we wish to speed-up design time, new tools are desired to manage their design at a higher abstraction level, ; designers should ignore what are the parts to be implemented in Hardware (HW) and what in Software (SW). Consequently, System-On-Chip (SoC) is revolutionizing the way electronic products are being developed. The higher electronic integration levels are creating new and user-friendly products with features that were not possible until recently. The range of possible target devices of your design is increasing. A full custom, mixed signal system-on-chip solution is extremely expensive and is only viable for high volume products such as mobile phones. However, a complete digital system can be integrated onto a single device economically using the new and very large field programmable gate arrays (FPGAs). The disadvantage of an FPGA implementation is that it will operate at a lower frequency than its equivalent application specific integrated circuit (ASIC). The trade-off of speed against using a commercial-off-the-shelf component is often acceptable.

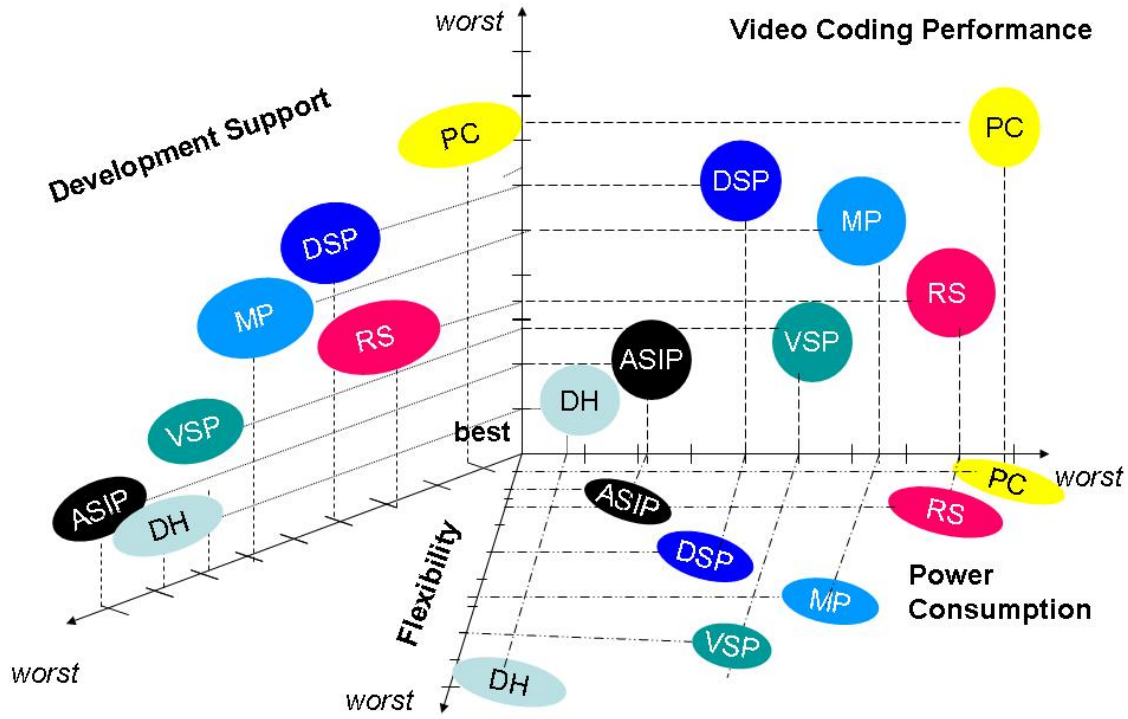


Figure 1.3: Comparison of platforms (approximate):DH: Dedicated Hardware, VSP: Video Signal Processor, RS: Reconfigurable System, PC, MP: Media Processor, ASIP: Application Specific Integrated Processor.

In a research environment, FPGAs are the ideal logic devices because of the immediate design implementation and the infinite re-programmability of static random access memory (SRAM) based components. Other possibility in the range of embedded electronic devices is the use of a DSP (Digital Signal Processors) or a DSP joint with a general processor. A more thoroughly explanation of the diverse trade-offs will be explained more truthfully in this thesis. It is a widespread well-known appreciation that performance has to be sacrificed for flexibility, and vice versa. Nevertheless, the gap between whole hardware solution (ASIC) and pure software execution (GPP General Purpose Processor). Nowadays is just in part addressed by domain specific processors such as DSPs or configurable logic. Consequently, not every required design objective is obliged to be met. If performance requirements of the application are not in the range of a factor 200-1000x lower that what can be done with achieved through hardware, then in most cases it is not possible to implement the task in software on a GPP.

- Bug fixes in software to avoid another re-spin of silicon.
- Posterior adaptation to standards.
- Multiple standards and market with similar HW platform.
- Means to advance in the performance of the solution after the silicon is transferred to the foundry.

Flexibility and performance are conflicting goals in today's solution space. This involves that adjustment on the functionality of the design can be done after the chip is taped out. A spectrum of solutions has been recognized in the past few years to target this gap. Figure 1.3 shows some target solutions that close the gap between GPP and HW (FPGA and ASIC) solutions. Such as: Processors extensions, Configurable (co)-processors, Programmable Accelerators and Reconfigurable Computing Engines are the most prominent current names employed in industry.

1.9.1 High Level Specification Languages

Designers have been using C and C++ for hardware design for many years. There is nothing new about that. What is new is the possibility of having synthesis and verification tools to obtain RTL description level. The need of HW-SW co-design languages for modeling IC and systems designs, either through C++ libraries like SystemC, systemVerilog or through abstractions like SpecC, extensions of C, like Handel-C. New languages like Esterel and ECL or higher levels of abstraction such as UML or JAVA. All these languages have in common that they support better reusable design descriptions. Giving us a single environment, which means we can model at multiple levels of abstraction. They have the capability to create objects that can build other objects. There's really no limit to the level of abstraction we can represent. Representing hardware concepts like parallelism will let us to develop rapid-substitution methods for the reuse of IP components and models.

Currently, it does not exist any language or Model of Computation (MoC) that serves for all purposes, neither in the near future it will exist. In this moment, every language tends to solve a given set of problems but leaves others without an easy solution. In the

next section, we explain very briefly some of them, with its pros and cons.

Our research group comes from the platform based design [65, 66, 67]. Therefore, we made a study about what specification language would be the most appropriate for our purposes. A language must have the possibility to describe systems at high level (event driven) or low level (RTL). Hardware and software descriptions must be feasible in the same language. UML [68] was considered since the core of this framework is a generic model of resource usage relationship that incorporates the notion of quality of services (QoS). The second sub-profile in this core framework contains a very general model of physical and measured time. Both continuous and discrete time models are supported. And parts of the profile are various: Timing mechanisms such as clocks and timers, which are common in most real-time operating systems. The third foundational element is a general model of concurrency and concurrency mechanisms. This includes the notion of active object (threads). UML was not used for this thesis due to my background was not the adequate for this language, and the language was not too much developed at the beginning of this work.

The mathematical language Matlab [69]. also was taken into account and its use was strongly proposed because it provides a complete environment for image acquisition, processing, visualization, analysis, and algorithm development. But as a weak point, it is still currently observed that it is not a co-design language since, we useSimulink tool to target the design to a FPGA (HW flow) or VisualDSP++ to obtain code software to be loaded on an embedded processor (SW flow). Therefore, both paths are not linked and there is not co-verification (results verification in the different intermediate design layers) path to silicon.

Another language that was regarded as co-design description languages are hardware description languages related with Java. Java uses a platform independent virtual machine. Adding hardware support for the virtual machine has been seen as one means to attain performance increase. This idea can be accomplished in either one of the three ways: (i) create a general microprocessor that is optimized for Java, yet still functions as a general processor; (ii) make a stand-alone Java processor that runs as a dedicated Java virtual machine; (iii) create a java coprocessor that works in unison with the general microprocessor. The main drawback of Java for Real Time Processing is that it is an interpreted language, so it is not appropriated if we expect a "real-time" response. There are basically two hardware description languages based on Java: Jbits and JHDL. J-Bits currently permits fast reconfiguration but a very low-level component library is still not available and this difficults development of complex systems. Currently in JBits research sphere, there is a main problem for a reasonable high-quality research developing. Jbits designers (basically Xilinx) do not want to open the details of how Xilinx FPGAs make the run-time bits-stream reconfiguration. Nowadays, run-time reconfiguration is still a concept to investigate for future electronics applications. Some good papers to go in deep with this language are [70, 71, 72, 73, 74].

JHDL [75] has been used as a co-design language. Some effort has been done in our research group to make Altera FPGAs programmable with JHDL, and itoffers nice properties for interactive multi-level and hardware-in-the-loop (HIL) verification [76, 61]. Maybe there is an opportunity with JHDL and NoC design since google has opened its

Android OS and the most standardized way to build applications is with Java [77, 78]. Concerning video applications, one of its main drawbacks is the lack of tools supporting real-time constrains. Also, JHDL implementation is mainly based on LPM libraries that are not supported in many implementation flows, although they are an EDIF standard.

Modeling with a C/C++ HW extension

One of the methods to design complex HW/SW systems is by adding an extension with hardware description primitives to a general programming language . More particularly, the method I selected for high level design description are C++ class libraries [79, 80, 81, 82]. Furthermore this can lead us to a unified environment where we can perform design space exploration, and mapping pieces of code down to hardware and software.

To support HW and MPSoC implementations, concurrency and reactivity have been added to C++ via class library extensions. And also a rich set of data types that permit the user to specify, for instance, fixed point signed and unsigned integers. The library also contains cycle accurate simulation kernel extensions that, along with the object-oriented mechanism built into C++, enable the design of hardware starting from C++ descriptions , and at a level of abstraction going from high-level mixtures of software and hardware down to cycle accurate synthesisable RTL. Since C++ was created as a general purpose programming language, this extended language is rich in both high level constructs.

There are different languages that appeared facing the design of complex systems. The first example of C++ extension for hardware description was SpecC, developed at UCI by David Gadsky's group [83, 84]. In parallel, EMP and UCB developed design flows (i.e. POLIS) from Esterel down to HW/SW platforms; Esterel is a language oriented to reactive (control) systems. It is not valid for data-flow systems. ECL is a language that has a sub-group of Esterel instructions with all the capabilities of C language. Another example is Cynlib (Open Source [85]). Ocapi-XL was also proposed (developed at IMEC Belgium). Finally, there is another one named handel-C developed by Celoxica -this is not C++ but just C. It is a language for implementing algorithm in hardware, architectural design space exploration, and hardware/software co-design. Based on ISO/ANSI-C, it has extensions required for hardware development. These include flexible data widths, parallel processing and communications between parallel elements. The language was designed around a simple timing model that makes it very accessible to system architects and software engineers.

C Extension for HW development: SystemVerilog and HandleC Language

The IEEE 1800TM SystemVerilog [86] is the industry's first unified hardware description and verification language (HDVL) standard. SystemVerilog is a major extension of the established IEEE 1364TM Verilog language. It was originally developed by Accellera for production in the design of large gate-count, IP-based, bus-intensive chips. SystemVerilog is targeted primarily at the chip implementation and verification flow, with powerful

links to the system-level design flow. SystemVerilog has the VHDL semantics that adds higher level data types and management functionality improving verilog 95 and 2001 that werejust for design and testbench with C features for software development.

In a similar way, handle-C [87] language paradigm is its ISO/ANSI-C based syntax with HW extension for a high level co-design framework that allows seamless integration of software and hardware design. This language is deprecated and currently most of the handle-C system designers have moved to SystemC.

C++ Extension for HW development: SystemC language

Last but not least, SystemC is an open source library in C++ for developing cycle-accurate or more abstract models of software, hardware architecture and system-level designs. SystemC is meant to be interoperable modeling platform allowing seamless tool integration. Current version of SystemC for HW implementation is v2.2. SystemC has a TLM library that allows Transaction-Level Modeling and also has a library for verification. Current work underway includes the gathering of requirements and development of consensus for implementation of temporal assertion support for SystemC. SystemC provides hardware-oriented constructs within the context of C++ as a class library implemented in standard C++. It uses spans design and verification from concept to implementation in hardware and software. SystemC provides an interoperable modeling platform which enables the development and exchange of very fast system-level C++ models. It also provides a stable platform for development of system-level tools. The Open SystemC Initiative (OSCI) is an independent not-for-profit organization composed of a broad range of companies, universities and individuals dedicated to supporting and advancing SystemC as an open source standard for system-level design. OSCI has released the AMS Draft 1 Standard introducing system-level design and modeling of embedded Analog/Mixed-Signal (AMS) systems using SystemC AMS extensions. SystemC release 3.0 is expected in the future and it is thought for embedded SW and RTOs implementations [81, 88]. SystemC seems to be the most common C++ library used for hardware description by Universities and Silicon Companies and a standard de facto. And also, It is has been recently (2007) accepted as a standard language by the IEEE. These are the reasons that all the development presented in this thesis is implemented with this language.

1.10 Thesis contributions and table of contents

The main motivation of our work is to compare the whole implementation process. Starting from the same reference specification (golden model), through different design methodologies (with their specific optimizations) down to different implementation platforms. This will let us to a fair comparison between platforms. Current approximations do not compare in a fair way (same technology 90nm, same original golden model). Another motivation is to propose a design methodology oriented to produce efficient and flexible implementations of a given algorithm described at system level that can later be

mapped into different architectures with intrinsically different power performance trade-offs. Another contribution is to observe this MPEG-4 MP IP as a Process Element (PE) connected to a bus or a NoC; this will allow scale results and will permit to increase the image size. All this can be realized making independent computation from communication to the largest possible extent. We will propose a communication type that can be extended to whole kind of Process Elements (of data-flow type). Response time (real-time) will be one of its requirements. Finally, we will show a NoC infrastructure for connecting several IPs . Until now it does not exist any work that, coming from an unique code, has been implemented in different platforms so that they can be compared for the hardware design of a complex IP (not only a series of loops). Getting comparative results among platforms will let us manage better the implementation decisions.

This thesis is divided in seven chapters and an appendix. In **chapter 1** there is an introduction to Architectures and Languages for HW-SW co-design. It focuses in taxonomy of platforms for multimedia. This chapter explains differences between single core and multi core, and homogeneous and heterogeneous processor platforms. It also explains platforms based in ISPs and custom platforms. ESL framework is presented and finally, there is the justification about the design language used for this research work. **Chapter 2** presents the video compression framework. It depicts the human visual system. Consequently, gives details how to attain quality measurements and about the most used standards for video compression as MPEG. Chapter gives main MPEG features providing a short introduction to the different standards from MPEG-1 to MPEG-21, passing through MPEG-2, 4 and 7. Afterwards, the main algorithms related with video VOP coding are explained. Chapter provides details about main algorithms for spatial and temporal coding.

In **chapter 3**, there is an introduction and state-of-the art to the different Multimedia platforms for MPEG compression. This chapter details the C++ work-flow model scheme proposed. There is also an explanation of sensor model, DMA and MPEG system implemented. As main focus, this chapter elucidate relatively accurate exploration and target platform comparison.

Chapter 4 focuses in customized platform architectures (FPGA and ASIC). Behavioral synthesis is shown. In both architectures, there is an explanation about the metrics used, mainly area and energy estimation models . It is also detailed the platform dependent optimizations. Two implementations are realized: one optimized for size and the other for speed . This is realized unrolling some internal loops in the code and therefore, using more FUs in parallel. We provide performance results and ASICs layout with 90nm technology.

Chapter 5 focuses in Instruction Set Processor Platforms ISP for a domain specific application like it is image compression (DSP and ASIP are our implemented solutions). We give details about frameworks, metrics and power models used. We provide an explanation of the different trade-offs and platform dependent optimizations. This optimizations permit increasing performance. This chapter provides also enlighted results and comparative tables among the fourth targeted platforms. Tables describe differences in terms of energy, execution time, area and design time. At the end of the chapter, there is an explanation about the comparison and a summary.

Chapter 6 considers MPEG compressor as an IP, that will be a PE of a bus or a NoC. Data-structures are combined in a way that each PE can work almost independently. A NoC was built in systemC and connected to the IP. On top of this model, DVFS techniques can be applied for QoS considerations, considering that current electronics systems do not have to work always in Worst Case Execution Time (WCET).

Chapter 7 details the main contributions of this thesis, final conclusions and future work. Finally, there is an **Appendix** with an explanation of the methodology followed to obtain Processor optimal points.

1.11 Summary

This chapter as already mentioned has explained the diverse possible architectures for the digital system platform implementation. It has proposed an architectural taxonomy for multimedia systems. It has also defined several platform concepts. Furthermore, it has clarified the varied ways of inter-tile communication. Then tile is established as an IP where the computation is carried out. Chapter defines a number of concepts such as: mono-core, multi-core –that bears associated other concepts such as: WLP, ILP and TLP. As well, it describes distinctions between a custom and a reconfigurable platform. The possibilities that exist for Instruction Set Processor solutions are : General Purpose Processor, Domain Specific Integrated Processor, General Digital Signal Processor, Media Processor, Video Processor and finally a Co-processor.

In the other way, it is presented the Custom Hardware Architectures and the reconfigurable architectures with its pros and cons. Next, in the chapter, it has justified the language employed in this thesis to describe the entire system. Finally, there is a table of contents with the contributions of this thesis.

2 The Video Compression Framework

2.1 The Human Visual System

A critical design goal for a digital video system is that the visual images produced by the system should be ´pleasing´ to the viewer. In order to achieve this goal it is necessary to take into account the response of the human visual system (HVS). The HVS is the ´system´ by which a human observer views, interprets and responds to visual stimuli. The main components of the HVS are shown in figure 2.1.

- *Eye*: The images focused by the *len*s onto the photodetecting area of the eye, the *retina*. Focusing and object tracking are achieved by muscles and the *iris* controls the aperture of the lens and hence the amount of light entering the eye.
- *Retina*: The retina consists of an array of *cones* (photoreceptors sensitive to colour at high light levels) and *rods* (photoreceptors sensitive to luminance at low light levels). The more sensitive cones are concentrated in a central region (the *fovea*) which means that high resolution colour vision is only achieved over a small area at the centre of the field of view.
- *Optic nerve:* This carries electrical signals from the retina to the brain.
- *Brain:* The human brain processes and interprets visual information, based partly on the received information (the image detected by the retina) and partly on prior learned responses (such as known objects shapes).

Diverse sorts of information are processed by different photosensitive cells in the retina of your eye, traveling to the brain combines signals from these visual channels to interpret the image. Light passing though the cornea is focused by the lens onto the retina, passing first through fine blood vessels, then through neural wiring to reach the photoreceptors then processes, encodes and ships off the resulting data to the brain via the optic nerve. The optical nerve terminates in the brain's lateral geniculate nucleus, a kind of switchboard that directs data else where into the brain for further processing.

The photoreceptors fall into two broad classes –rods for night vision and cones for high-resolution color vision. The rods provide high gain at the cost of low resolution, in part because each rod is connected to many cells in the optic nerve. The cons resolve fine details, because each one is connected through an intermediate layer of image processing nerves to a single optic nerve cell.

For the sensors to work properly, the light hitting them must vary continually, so as to refresh the signal. The eye performs such refreshing, even on static images, by making slight intermittent movements, called saccades.

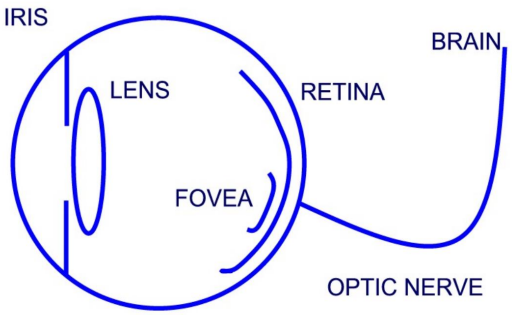


Figure 2.1: HVS Components

Half the cones are in the fovea, a 2-millimeter area in the macula. It spans two degrees of the visual field; of these cones, half fall in the very center, spanning 0.5 degree. Cones come in varieties sensitive to long, medium, and short wavelength, corresponding roughly to red, green and blue, respectively. There are roughly 30 times as many red- and green-sensitive cones as blue- sensitive cones.

The red and green cones perceive the range of wavelengths between 500 and 700 nanometers on the visible light spectrum. The human eye tends to focus on yellow (580nm), because that wavelength gives the best overall image quality, exciting the green and red cones nearly equally. Far-red and far-blue wavelengths provide little resolution.

Output from the cones undergoes preprocessing in neural layers in the retina, right and above the cones. The neural layers encode an image into a luminance channel, which distinguish black from white, and two chrominance channels, one managing red versus green, and the other, yellow versus blue.

These preprocessing neural layers generate the luminance channel by combining inputs from the red cones and the green cones and effectively ignoring the blue cones. They generate the red/green chroma channel by subtracting green-cone input from red-cone input. The yellow/blue chroma channel subtracts the blue-cone inputs from those of the red and green cones. The operation of the HVS is a large and complex area of study. Some of the important features of the HVS that have implications for digital video design are listed in table 2.1.

2.2 Video Quality

In order to specify, evaluate and compare video communication systems it is necessary to determine the quality of the video images displayed to the viewer. Measuring visual quality is a difficult and often imprecise art because of the big of factors that may influence the results. Visual quality is inherently *subjective* and is therefore influenced by many subjective factors that difficult a completely accurate quality measurement.

Features	Implication for digital video systems
The illusion of smooth motion can be achieved by presenting a series of images at a rate of 20-30Hz or more	Video systems should aim for frame repetition rates of 20 Hz or more for natural moving video
HVS responses vary from individual to individual	Multiple observers should be used to assess the quality of a video system
The HVS is more sensitive to luminance than to color detail.	Colour (chrominance) resolution may be reduced without significantly affecting image quality.
The HVS is more sensitive to high contrast (i.e. large differences) in luminance than low contrast	Large changes in luminance (e.g. edges in an image) are particularly important to appearance of the image
The HVS is more sensitive to low spatial frequencies (i.e. changes in luminance that occur over a large area) than high spatial frequencies (rapid changes that occur in small area)	It may be possible to compress images by discarding some of the less important higher frequencies (however, edge information should be preserved)
The HVS is more sensitive to image features that persist for a long duration.	It is important to minimize temporally persistent disturbances or artefacts in an image

Table 2.1: Features of the Human Visual System

Measuring visual quality using objective criteria gives accurate, repeatable results, but as yet there are no objective measurement systems that will completely reproduce the subjective experience of a human observer watching a video display.

2.2.1 Objective Quality Measurement

Because of the problems of subjective measurement, developers of digital video systems rely heavily on objective measures of visual quality. Objective measures have not yet replaced subjective testing: however, they are considerably easier to apply and are particularly useful during development and for comparison purposes.

Probably the most widely used objective measure is peak signal to noise ratio (PSNR), calculated using equation 2.1. PSNR is measured on a logarithmic scale and is based on the mean squared error (MSE) between an original and an impaired image or video frame, mean square error, relative to $(2^n - 1)^2$ (the square of the highest possible signal value in the image). PSNR can be calculated very easily and is therefore a very popular quality measure. It is widely used as a method of comparing the quality of compressed and decompressed versions of the original image.

$$PSNR_{dB} = 10 \cdot log_{10} \frac{(2^n - 1)^2}{MSE} \tag{2.1}$$

The most traditional ways of evaluating quality of digital video processing systems (e.g. video codec like DivX, XviD) are calculation of the signal-to-noise ratio (SNR) and peak signal-to-noise ratio (PSNR) between the original video signal and signal passed through this system. PSNR is the most widely used objective video quality metric. However, PSNR values do not perfectly correlate with a perceived visual quality due to non-linear behavior of human visual system. Recently a number of more complicated and precise metrics were developed, for example UQI, VQM, PEVQ, SSIM and CZD. Based on a benchmark by the Video Quality Experts Group (VQEG) in the course of the Multimedia Test Phase 2007-2008 some metrics were standardized as ITU-T Rec. J.246 (RR) and J.247 (FR) in 2008 [89, 90, 91].

The performances of an objective video quality metric are evaluated by computing the correlation between the objective scores and the subjective tests results. The latters are called mean opinion score (MOS). The most frequently used correlation coefficients are : linear correlation coefficient, Spearman's rank correlation coefficient, kurtosis, kappa coefficient and outliers ratio.

When estimating quality of a video codec, all the mentioned objective methods may require repeating post-encoding tests in order to determine the encoding parameters that satisfy a required level of visual quality, making them time consuming, complex and impractical for implementation in real commercial applications. For this reason, a lot of research has been focused on developing novel objective evaluation methods which enable prediction of the perceived quality level of the encoded video before the actual encoding is performed [92].

In the following partdifferent MPEG standard versions are introduced.

2.2.2 The MPEG VIDEO Compression Standard

The Motion Picture Experts' Group was assembled by the International Standard Organization (ISO) to establish standards for the compression, encoding, and decompression of motion video. MPEG-1 [93] is a standard supporting compression of image resolutions of approximately 352x288 at 30 fps into a data stream of 1.5Mbps. This data rate is suitable for pressing onto CD-ROM. The MPEG-2 standard [94] supports compression of broadcast television (705x576 at 30fps) and HDTV (1920x1152 at 60fps) of up to 60 Mpixels/sec (approx. 700Mb) at compression ratios of roughly three times those expected of moving JPEG [95] (a playback rates of up to 80Mbps).

The MPEG standard specifies the functional organization of a decoder. The data stream is cached in a buffer to reduce the effect of jitter in delivery and decode, and is demultiplexed into a video stream, an audio stream, and additional user-defined streams. The video stream is decoded into a "video sequence" composed of the sequence header and groups of pictures.

2.2.2.1 MPEG-1 video History

Developments in video conferencing techniques and standards such as H.261 in the late 1980s formed the bases of the MPEG-1 standard in the early 1990s [96]. The Moving Picture Experts Group (MPEG) investigated how to store compressed video on a CD-ROM, which led to the development of the MPEG-1 standard.

MPEG-1 video was originally designed with a goal of achieving acceptable video quality at 1.5 Mbit/s data rates and 352x240 (29.97 frame per second) / 352x288 (25 frame per second) resolution. While MPEG-1 applications are often low resolution and low bitrate, the standard allows any resolution less than 4095x4095. Nevertheless, most implementations were designed with the Constrained Parameter Bitstream specification in mind. At present MPEG-1 is the most compatible format in the MPEG family; it is playable in almost all computers and VCD/DVD players. One big disadvantage of MPEG-1 video is that it supports only progressive pictures. This deficiency helped prompt development of the more advanced MPEG-2.

MPEG-1 defines a group of Audio and Video (AV) coding and compression standards agreed upon by MPEG (Moving Picture Experts Group). MPEG-1 video is used by the Video CD (VCD) format and less commonly by the DVD-Video format. The quality at standard VCD resolution and bitrate is near the quality and performance of a VHS tape. MPEG-1 Audio Layer 3 is the popular audio format known as MP3. As cheaper and more powerful consumer decoding hardware became available, more advanced formats such as MPEG-2 and MPEG-4 were developed. These newer formats are more complex and require more powerful hardware, but the formats also achieve greater coding efficiency, i.e., quality per bitrate.

MPEG-1 consists of several parts or steps, as follows:

1. Synchronization and multiplexing of video and audio (MPEG-1 Program Stream).
2. Compression codec for non-interlaced video signals.
3. Compression codec for perceptual coding of audio signals. The standard defines

three "layers," or levels of complexity, of MPEG audio coding.

1. MP1 or MPEG-1 Part 3 Layer 1 (MPEG-1 Audio Layer I)
2. MP2 or MPEG-1 Part 3 Layer 2 (MPEG-1 Audio Layer II)
3. MP3 or MPEG-1 Part 3 Layer 3 (MPEG-1 Audio Layer III)
4. Procedures for testing conformance.
5. Reference software.

2.2.2.2 Introduction to MPEG-2 standard

MPEG-2 is a standard for "the generic coding of moving pictures and associated audio information[94]." It describes a combination of lossy video compression and lossy audio compression (audio data compression) methods which permit storage and transmission of movies using currently available storage media and transmission bandwidth.

2.2.2.3 Main characteristics

It is commonly used around the world to specify the format of the digital television signals that are broadcast by terrestrial (over-the-air), cable, and direct broadcast satellite TV systems. It also specifies the format of movies and other programs that are distributed on DVD and similar disks. The standard allows text and other data, e.g., a program guide for TV viewers, to be added to the video and audio data streams. TV stations, TV receivers, DVD players, and other equipment are all designed to this standard. MPEG-2 was the second of several standards developed by the Motion Pictures Expert Group (MPEG) and is an international standard (ISO/IEC 13818). Parts 1 and 2 of MPEG-2 were developed in a joint collaborative team with ITU-T, and they have a respective catalog number in the ITU-T Recommendation Series.

While MPEG-2 is the core of most digital television and DVD formats, it does not completely specify them. Regional institutions adapt it to their needs by restricting and augmenting aspects of the standard. See tables 2.2 2.3 2.4 (about Profiles and Levels).

MPEG-2 Profiles					
Abbr.	Name	Frames	YCbCr	Streams	Comment
SP	Simple Profile	P, I	4:2:0	1	no interlacing
MP	Main Profile	P, I, B	4:2:0	1	
422P	4:2:2 Profile	P, I, B	4:2:2	1	
SNR	SNR Profile	P, I, B	4:2:0	1-2	SNR: Signal to Noise Ratio
SP	Spatial Profile	P, I, B	4:2:0	1-3	low, normal and HQ decoding
HP	High Profile	P, I, B	4:2:2	1-3	

Table 2.2: MPEG-2 Profiles and Levels (Profiles)

MPEG-2 includes a Systems part (part 1) that defines two distinct (but related) container formats. One is Transport Stream, which is designed to carry digital video and audio over somewhat-unreliable media. MPEG-2 Transport Stream is commonly

MPEG-2 Levels					
Abbr	Name	Pixel/line	Lines	Frame rate (Hz)	Bitrate (Mbit/s)
LL	Low Level	352	288	30	4
ML	Main Level	720	576	30	15
H-14	High 1440	1440	1152	30	60
HL	High Level	1920	1152	30	80

Table 2.3: MPEG-2 Profiles and Levels (Levels)

used in broadcast applications, such as ATSC and DVB. MPEG-2 Systems also defines Program Stream, a container format that is designed for reasonably reliable media such as disks. MPEG-2 Program Stream is used in the DVD and SVCD standards. MPEG-2/System is formally known as ISO/IEC 13818-1 and as ITU-T Rec. H.222.0 [97].

The Video part (part 2) of MPEG-2 is similar to MPEG-1, but also provides support for interlaced video (the format used by analog broadcast TV systems). MPEG-2 video is not optimized for low bit-rates (less than 1 Mbit/s), but outperforms MPEG-1 at 3 Mbit/s and above. All standards-conforming MPEG-2 Video decoders are fully capable of playing back MPEG-1 Video streams. MPEG-2/Video is formally known as ISO/IEC 13818-2 and as ITU-T Rec. H.262 [94] . With some enhancements, MPEG-2 Video and Systems are also used in most HDTV transmission systems.

The MPEG-2 Audio part (defined in Part 3 of the standard) enhances MPEG-1's audio by allowing the coding of audio programs with more than two channels. Part 3 of the standard allows this to be done in a backwards compatible way, allowing MPEG-1 audio decoders to decode the two main stereo components of the presentation.

Part 7 of the MPEG-2 standard specifies a rather different, non-backwards-compatible audio format. Part 7 is referred to as MPEG-2 AAC. While AAC is more efficient than the previous MPEG audio standards, it is much more complex to implement, and somewhat more powerful hardware is needed for encoding and decoding.

2.2.2.4 Video coding scheme

This part just explains a simplified view of how to code video. All the information can be obtained from the different references. You must think that an HDTV camera generates a raw video stream of more than one billion bits per second. This stream must be compressed if digital TV has to fit in the available bandwidth assigned to TV channels (depending on the distribution channel) and if movies are to fit on DVDs. Fortunately, video compression is practical because pictures data are often redundant in space and time. For example, the sky can be blue across the top of a picture, and that blue sky can persist for several frames. Also, according to the way our eyes work, it is possible to delete some data from video pictures with almost no noticeable degradation in image quality. TV cameras used in broadcasting usually generate 50 pictures a second (in Europe and elsewhere) or 59.94 pictures a second (in North America and elsewhere). Digital television requires these pictures to be digitized so that they can be processed by

Profile@ Level	**Resolution (px)**	**Frame-rate(Hz)**	**Sampling**	**Bitrate (Mbit/s)**	**Example Application**
SP@LL	176 × 144	15	4:2:0	0.096	Wireless handsets
SP@ML	352 × 288	15	4:2:0	0.384	PDAs
	320 × 240	24			
MP@LL	352 × 288	30	4:2:0	4	Set-top boxes (STB)
MP@ML	720 × 480	30	4:2:0	15(1*)	DVD(2*), SD-DVB(1*)
	720 × 576	25		9.8 (2*)	
MP@H-14	1440 × 1080	30	4:2:0	80	(HDV: 25) HDV
	1280 × 720	60			
MP@HL	1920 × 1080	30	4:2:0	80	ATSC 1080i, 720p60,
	1280 × 720	30			HD-DVB (HDTV)
422P@LL			4:2:2		
422P@ML	720 × 480	30	4:2:2	50	Sony IMX using I-frame only,
	720 × 576	25			Broadcast video
422P@H-14	1440 × 1080	30	4:2:2	80	Future MPEG-2-based HD
	1280 × 720	60			products
422P@HL	1920 × 1080	30	4:2:2	300	Future MPEG-2-based HD
	1280 × 720	60			products

Table 2.4: MPEG-2 Profiles and Levels (Profiles @ Levels)

computer hardware. Each picture element (a pixel) is then represented by one luminance number and two chrominance numbers. These describe the brightness and the color of the pixel (YCbCr). Thus, each digitized picture is initially represented by three rectangular arrays of numbers.

A common method (herited from analog TV) to reduce the amount of data managed is to separate the picture into two fields: the "top field," which is the odd numbered rows, and the "bottom field," which is the even numbered rows. The two fields are displayed alternately. This is called interlaced video. Two successive fields are called a frame. The typical frame rate is then 25 or 29.97 frames a second. Not interlaced video is called progressive video and each picture is a frame. MPEG-2 supports both options.

Another technique to reduce data rate is to thin out the two chrominance matrices. The remaining chrominance values represent the nearby values that are deleted. Thinning works because the eye is more responsive to brightness than to color. The 4:2:2 chrominance format indicates that half the chrominance values have been deleted. The 4:2:0 chrominance format indicates that three quarters of the chrominance values have been deleted. If no chrominance values have been deleted, the chrominance format is 4:4:4. MPEG-2 allows all three options.

MPEG-2 specifies that raw frames have to be compressed into three kinds of frames: intra-coded frames (I-frames), predictive-coded frames (P-frames), and bidirectionally-predictive-coded frames (B-frames).

An I-frame is a compressed version of a single uncompressed (raw) frame. It takes

advantage of spatial redundancy and of the inability of the eye to detect certain changes in the image. Unlike P-frames and B-frames, I-frames do not depend on data in the preceding or following frames. Temporarily, the raw frame is divided into 8 pixel by 8 pixel blocks[98]. Data in each block is transformed by a discrete cosine transform. The result is an 8 by 8 matrix of coefficients. The transform converts spatial variations into frequency variations, but it does not change the information in the block; the original block can be recreated exactly by applying the inverse cosine transform. The advantage of doing this is that the image can now be simplified by quantizing coefficients. Many of the coefficients, usually the higher frequency components, will then be zero. The penalty of this step is the loss of some subtle distinctions in brightness and color. If one applies the inverse transform to the matrix after it is quantized, one gets an image that looks very similar to the original image but that is not quite as nuanced. Next, the quantized coefficient matrix is itself compressed. Typically, one corner of the quantized matrix is filled with zeros. By starting in the opposite corner of the matrix, then zigzagging through the matrix to combine the coefficients into a string, then substituting run-length codes for consecutive zeros in that string, and then applying Huffman coding to that result, one reduces the matrix to a smaller array of numbers. This array is broadcast or put on DVDs. In the receiver or the player, the whole process is reversed, enabling the receiver to reconstruct, to a close approximation, the original frame. This part will be explained in the next section where it is explained each algorithm more accurately.

Typically (not for a real time compression), every Nth frame or so is made into an I-frame. P-frames and B-frames might follow an I-frame like this, IBBPBBPBBPBB(I), to form a Group Of Pictures (GOP). However, the MPEG standard is flexible about this. Reader should take into account that this GOP is not for real-time compression since the amount of memory and the computation necessary to compute the GOP would be too high.

P-frames provide more compression than I-frames because they take advantage of the data in the previous I-frame or P-frame. I-frames and P-frames are called reference frames. To generate a P-frame, the previous reference frame is reconstructed, just as it would be in a TV receiver or DVD player. The frame being compressed is divided into 16 pixel by 16 pixel macro blocks. Then, for each of those macro blocks, the reconstructed reference frame is searched to find the 16 by 16 macro block that best matches the macro block being compressed. The offset is encoded as a "motion vector." Frequently, the offset is zero. But, if something in the picture is moving, the offset might be something like 23 pixels to the right and 4 pixels up. The match between the two macro blocks will often not be perfect. To correct this, the encoder computes the strings of coefficient values as described above for both macro blocks and, then, subtracts one from the other. This "residual" is appended to the motion vector and the result sent to the receiver or stored on the DVD for each macro block being compressed. Sometimes no suitable match is found. Then, the macro block is treated like an I-frame macro block.

The processing of B-frames is similar to that of P-frames except that B-frames use the picture in the following reference frame as well as the picture in the preceding reference

frame. As a result, B-frames usually provide more compression than P-frames. B-frames are never reference frames. While the above generally describes MPEG-2 video compression, there are many details that are not discussed including details involving fields, chrominance formats, responses to scene changes, special codes that label the parts of the bitstream, and other pieces of information. This section is explained more thoroughly in [99] and in section "MPEG Encoder Decoder" of this chapter.

2.2.2.5 MPEG-2 Audio encoding

We have not implemented any algorithm for audio encoding. But the standard explains how to develop them. MPEG-2 also introduces new audio encoding methods but this thesis just shows video coding information so I will just mention some information about audio. These are:

- low bitrate encoding with halved sampling rate (MPEG-1 Layer 1/2/3 LSF)
- multichannel encoding with up to 5.1 channels
- MPEG-2 AAC

2.2.3 Introduction to MPEG-3

MPEG-3 is the designation for a group of audio and video coding standards agreed upon by MPEG (Moving Picture Experts Group). MPEG-3 was designed to handle HDTV signals in the range of 20 to 40 Mbit/s. It was soon discovered that similar results could be obtained through slight modifications to the standard MPEG-2 . Shortly thereafter, work on MPEG-3 was discontinued. MPEG-3 should not be confused with MPEG-1 Part 3 Layer 3 (or MPEG-1 Audio Layer 3), commonly referred as MP3.

2.2.4 Introduction to MPEG-4

MPEG-4 is a standard used primarily to compress audio and visual (AV) digital data. Introduced in late 1998, it is the designation for a group of audio and video coding standards and related technology agreed upon by the ISO/IEC Moving Picture Experts Group (MPEG) under the formal standard ISO/IEC 14496. The uses for the MPEG-4 standard are web (streaming media) and CD distribution, conversation (videophone), and broadcast television, all of which benefit from compressing the AV stream.

MPEG-4 absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, adding new features such as (extended) VRML support for 3D rendering, object-oriented composite files (including audio, video and VRML objects), support for externally-specified Digital Rights Management and various types of interactivity. AAC (Advanced Audio Codec) was standardized as an adjunct to MPEG-2 (as Part 7) before MPEG-4 was issued.

MPEG-4 is still a standard under development and is divided into a number of parts. Unfortunately the companies promoting MPEG-4 compatibility do not always clearly state which "part" level compatibility they cover. The key parts to be aware of are MPEG-4 part 2 (MPEG-4 SP/ASP, used by codecs such as DivX, XviD and 3ivx and by Quicktime 6) and MPEG-4 part 10 (MPEG-4 AVC/H.264, used by the x264 codec,

by Quicktime 7, and by new-gen DVD formats like HD DVD -already deprecated- and Blu-ray disc).

Most of the features included in MPEG-4 are left to individual developers to decide whether to implement them. This means that there are probably no complete implementations of the entire MPEG-4 set of standards. To deal with this, the standard includes the concept of "profiles" and "levels", allowing a specific set of capabilities to be defined in a manner appropriate for a subset of applications. More information can be obtained in [100, 101].

One of the key contributions of MPEG-4 Visual is a move away from the "traditional" view of a video sequence as being merely a collection of rectangular frames of video. Instead, MPEG-4 Visual treats a video sequence as a collection of one or more *video objects*. MPEG-4 Visual defines a video object as a flexible "entity that a user is allowed to access (seek, browse) and manipulate (cut, and paste)" [102]. A video object (VO) is an area of the video scene that may occupy an arbitrary-shaped region and may exist for an arbitrary length of time. An instance of a VO at a particular point in time is a video object plane (VOP). This definition encompasses the traditional approach of coding complete frames, in which each VOP is a single frame of a video and a sequence of frames forms a VO. For example, figure 2.2 shows a VO consisting of three rectangular VOPs. However, the introduction of the VO concepts allows more flexible options for coding frame and each one is coded separately (object-based coding). In this manuscript, there is the approximation that a video object plane (VOP) is a complete rectangular image, therefore during the reading a VO is similar to a GOP (Group of Pictures).

2.2.5 Introduction to MPEG-7

MPEG-7 is a multimedia content description standard. This description will be associated with the content itself, to allow fast and efficient searching for material that is of interest to the user. MPEG-7 is formally called Multimedia Content Description Interface. Thus, it is not a standard which deals with the actual encoding of moving pictures and audio, like MPEG-1, MPEG-2 and MPEG-4. It uses XML to store meta data, and can be attached to time code in order to tag particular events, or synchronize lyrics to a song, for example.

It was designed to standardize:

- a set of description schemes and descriptors (XML).
- a language to specify these schemes, called the Description Definition Language (DDL).
- a scheme for coding the description.

The combination of MPEG-4 and MPEG-7 has been referred to as MPEG-47.

MPEG-7 is intended to provide complementary functionality to the previous MPEG standards, by representing information about the content, not the content itself ("the bits about the bits"). This functionality is the standardization of multimedia content descriptions. MPEG-7 can be used independently of the other MPEG standards - the description might even be attached to an analog movie. The representation that is defined within MPEG-4, i.e. the representation of audio-visual data in terms of objects, is

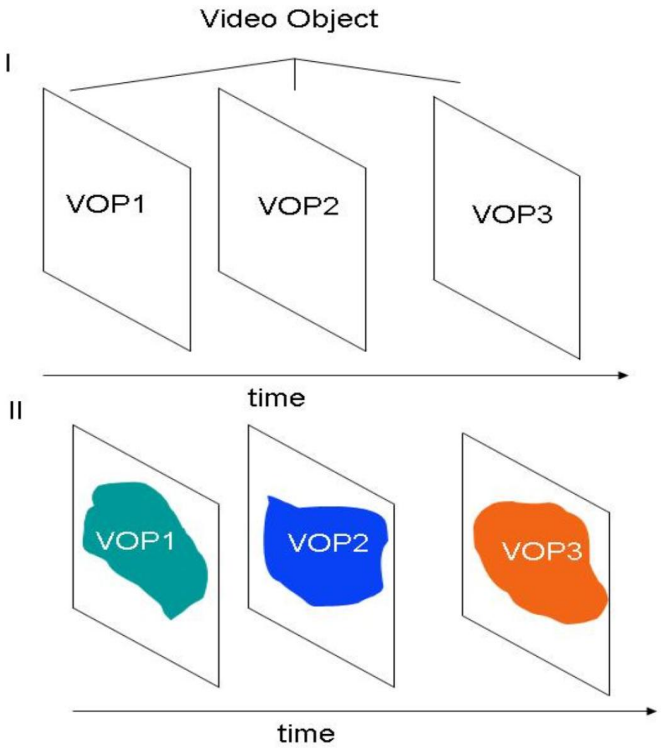


Figure 2.2: (I) VOPs and VO (rectangular). (II) VOPs and VO (arbitrary shape)

however very well suited to what will be built on the MPEG-7 standard. This representation is basic to the process of categorization. In addition, MPEG-7 descriptions could be used to improve the functionality of previous MPEG standards.

Objectives

- Provide a fast and efficient searching, filtering and content identification method.
- Describe main issues about the content (low-level characteristics, structure, models, collections, etc.).
- Index a big range of applications.
- Audiovisual information items that MPEG-7 deals are : Audio, voice, video, images, graphs and 3D models
- Inform about how objects are combined in a scene.

More information in [103, 104, 105, 106].

2.2.6 MPEG-21 Multimedia Framework

Based on the above observations, MPEG-21[**107**] aims at defining a normative open framework for multimedia delivery and consumption for use by all the players in the delivery and consumption chain. This open framework will provide content creators,

producers, distributors and service providers with equal opportunities in the MPEG-21 enabled open market. This will also be to the benefit of the content consumer providing them an access to a large variety of content in an interoperable manner.

MPEG-21 is based on two essential concepts: the definition of a fundamental unit of distribution and transaction (the Digital Item) and the concept of Users interacting with Digital Items. Digital Items can be considered the "what" of the Multimedia Framework (e.g., a video collection, a music album) and Users can be considered the "who" of the Multimedia Framework.

The goal of MPEG-21 can thus be rephrased to: defining the technology needed to support Users to exchange, access, consume, trade and otherwise manipulate Digital Items in an efficient, transparent and interoperable way.

During the MPEG-21 standardization process, Calls for Proposals based upon requirements have been and continue to be issued by MPEG. Eventually the responses to the calls result in different parts of the MPEG-21 standard (i.e. ISO/IEC 21000-N) after intensive discussion, consultation and harmonization efforts between MPEG experts, representatives of industry and other standards bodies (i. e. W3C, OMA, DLNA).

MPEG-21 identifies and defines the mechanisms and elements needed to support the multimedia delivery chain as described above as well as the relationships between and the operations supported by them. Within the parts of MPEG-21, these elements are elaborated by defining the syntax and semantics of their characteristics, such as interfaces to the elements.

2.3 MPEG-A: Multimedia Application Formats

The ISO/IEC 23000 standard, or MPEG-A[108], is a recent addition to the well-known standards developed by the Moving Picture Experts Group (MPEG; **[107])**, a working group of the International Organization for

Standardization/International Electrotechnical Commission (ISO/IEC). The goal for MPEG-A is to facilitate the swift development of innovative, standards-based multimedia applications and services. To meet this goal, the MPEG-A standard specifies multimedia application formats (MAFs) and provides the related software implementation.

The software demonstrates how MAFs are used and offers vendors a head start for developing multimedia products. MPEG's ultimate objective for MAFs is to stimulate the increased use of MPEG technology through additional interoperability of different media at the application level. Although MPEG has recommended profiles— that is, subsets of its technologies—for specific parts of the standards, such as MPEG-2 Video, it has never recommended the use of specific combinations of profiles across different parts of a standard, such as MPEG-2 Audio and MPEG-2 Video, or across different standards, such as MPEG-4 and MPEG-7. Instead, groups such as the Digital Video Broadcasting Project, 3rd Generation Partnership Project, and Internet Streaming Media Alliance have created their own application-specific combinations of technologies. The combination of technologies resulting from this process, although tightly matched to the needs of specific application domains, does not necessarily allow multimedia for-

mats to be interoperable across different market segments, because each of those groups happens to choose a combination of standardized components such that the end results differ enough to prevent them from being interoperable. With MPEG-A and its MAFs, MPEG has taken on the task of packaging technology to determine complete formats for multimedia data, which will result in generically usable specifications and, consequently, a greater degree of interoperability. This enhanced interoperability will benefit many different businesses that are engaged in dealing with digital content.

2.4 MPEG Encoder and Decoder

This section is focused in the kernel (as will be explained in chapter 3 and 4; kernel is the most data-flow part that include ME, MC and DCT) algorithms developed in this thesis. The specification of the MPEG encoder defines many compression options. While all of these options must be supported by the decoder, the selection of which options have to support in compression is left to the discretion of the implementer. An MPEG encoder may choose compression options balancing the need for high compression ratios against the complexity of motion compensation or adaptive quantization calculations. Decisions will be affected by such factors as:

- A need for real-time compression, MPEG algorithms are complex, and there may be sufficient time to implement singular options on a particular platform.
- A need for high compression ratios. For highest possible compression ratios at highest possible quality, every available option must be exercised.
- A need for insensitivity to transmission error. MPEG-2 and MPEG-4 supports recovery from transmission errors. Some error recovery mechanism are implemented by the decoder.
- Fast algorithms. Development of fast algorithms may make compression options available that would otherwise be impractical.
- Availability of special hardware. Dedicated hardware may increase the performance of the encoder to the point that additional compression options can be considered.

The MPEG visual coding is designed for natural images. Natural images have two central features.

- They are spatially smooth. Abrupt changes are unusual, and mainly due to objects boundaries.
- They are temporally smooth. Natural objects do not change or move suddenly.

These two assumptions are exploited in MPEG coding using two algorithms:

- Frequency coding (Spatial): image is transformed to a frequency domain using the DCT transform. In this domain, the components corresponding to high frequencies are unusual. This type of coding is also called intra coding, because it does not depend on data from other frames.
- Predictive coding (Temporal): The contents of a frame are derived from the contents of previous (and future) frames, plus an error, which is also frequency encoded and is usually very low.

MPEG frames are divided into 16x16 macro blocks (MB), which are further divided into 8x8 blocks (four for the luminance channel, and one for each downsampled chrominance 4:2:0). Each MB can be either frequential or predictive coding , depending on what type produces fewer bits. We differentiate between three types of picture depending on how MBs are encoded:

- Intra-coded VOB Video Object (I-VOB): All the MBs are intra-coded.
- Predictive-coded VOB (P-VOB): MB can be predictive coding using previous frame.
- Bidirectional-coded VOB (B-VOB): MBs can be predictive coded using a previous frame, a future frame, or both.

P and B VOB frames may also contain intra-coded MBs. P and B frame require reference frames from where their content are inferred. Only I and P VOB frames are allowed as reference frames for predictive coding. An example of MPEG picture type sequence an their dependence is shown in figure 2.3. Reference frame must be always available before the pictures that depend on them. For this reason the order of transmission of picture (or bitstream order) is different from their display order. Therefore, the sequence of transmission of the picture would be IPBB instead of the original display order sequence IBBP. The MPEG decoder has two pictures. Note that the P VOB picture can be decoded because it only depends on the previous picture reference picture, which is the previous I frame. When a new reference picture is received, the previous backward reference is displayed and replaces the previous forward reference in its buffer. The new reference picture fills the backward reference buffer. In the MPEG standard, frames in a sequence are coded using three different algorithms, as illustrated in Figure 2.3:

I VOB frames (intra-frames) are self-contained and coded using a DCT-based technique similar to JPEG. I VOB frames are used as random access point in MPEG streams, and they give the lowest compression ratios within kind of MPEG frames. P VOB frames (predicted frames) are coded using forward predictive coding. The compression ratios of P VOB frames is significantly higher than the I frames. B frames (Bidirectional or interpolated frames) are coded using two reference frames, a past and a future frame (which can be I or P frames). Bidirectional, or interpolated coding provides the highest amount of compression [109].

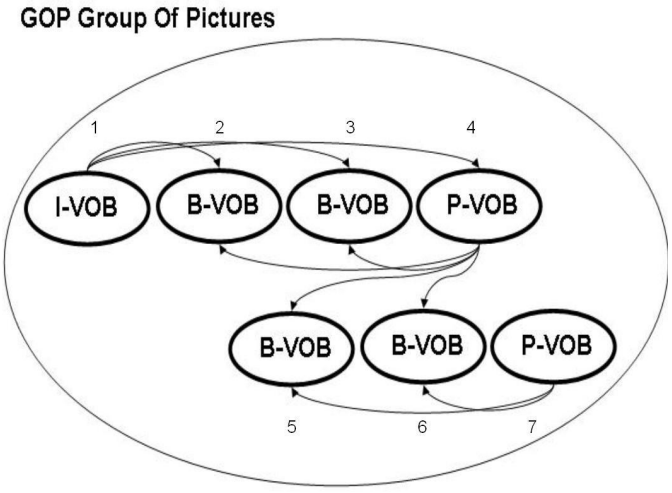


Figure 2.3: Video Object VO: Formed by 1 I Visual Object (VOB), 2 Predictive Visual Objects and 4 Bi-directional Video Objects.

I, P and B VOP frames are described in more detail in the following sections. Note that in Figure 2.3 the first two B frames (2 and 3) are bidirectionally coded using the past frame I (frame 1), and the future frame P (frame 4). Therefore, the decoding order will differ from the encoding order. The P VOB frame 4 must be decoded before B VOB frames 2 and 3 and P VOB frame 7 before frames 5 and 6. If the sequence is transmitted over a channel (i.e: network), the actual transmission order should be {1,4,2,3,7,5,6}.

The MPEG application determines a sequence of I, P, and B VOB frames. If there is a need for fast random access, the best resolution would be achieved by coding the whole sequence as I VOB frames (MPEG becomes identical to Motion JPEG). However the highest compression ratios can be achieved by incorporating a large number of B VOB frames.

2.5 MPEG Data Stream

The MPEG specifications define a "video sequence" composed of a video sequence header and many Group-of-Pictures or VOs see figure 2.3. The video sequence header defines the video format, picture dimensions, aspect ratio, frame rate, and delivered data rate. Supported video format include ITU-R BT.601, HDTV(16:9), and VGA. Supported chroma formats include "4:2:0"(YUV) and "4:4:4"(RGB). A suggested buffer size for the video sequence is also specified, a number intended to buffer jitter caused by differences in decode time.

A VO contains pictures that may be encoded into one of three supported compression formats. The VO header contains a starting time for the group, and can therefore be used as a point of random access. Each frame within the VO is numbered, and its

number coupled with the VO start time and the playback frame rate determines its playback time. Each picture is subdivided into "slices" and then into "macro blocks", as already explained in subsection 2.2.2.4. A macroblock is composed normally of four 8x8 pixels or blocks of luminance data, and typically two 8x8 blocks of chrominance data, one Cr and one Cb for the 4:2:0 format.

2.5.1 I Video Object Picture format

The I (Intraframe) VOB Picture format substantially correspond to the JPEG format. These pictures are encoded by transformation into DCT space, quantization of the resultant coefficients, and entropy coding of the result. Transformation into DCT space is performed by an 8x8 DCT. Quantization is performed by reference to a user-loadable quantization table modified by a scale factor. This mechanism supports adaptive quantization are the cost of additional complexity - although 30% improvement in compression is claimed [95]. After quantization, the resulting coefficients are reordered in zigzag order, run-length coded, variable-length coded, and entropy coded. The resulting data stream should show roughly JPEG levels of compression.

2.5.2 P Video Object Picture format

The P (Predicted) VOB Picture format introduces the concept of motion compensation. Each macro block is coded with a vector that predicts its value from an earlier I or P frame. The decoding process copies the contents of the macro block-sized data at the address referenced by the vector into the macro block of the P frame currently being decoded. Five bits of resolution are reserved for the magnitude of the vector in each of the x and y directions meaning that 1024 possible data blocks may be referenced by the predicted macro block. However, eight possible magnitude ranges may be assigned to those five bits, meaning as many as 8192 macro blocks might have to be evaluated to exhaustively determine the best vector. Each evaluation might require testing as many as 384 pixels, and a further complexity is seen as performing fractional interpolation of pixels (vectors motion as small as 1/2 pixels are supported). Finally the difference between the prediction and the macro block to be compressed may be encoded in like fashion to I frame encoding above.

2.5.3 B Video Object Picture format

The B (Bidirectional prediction) VOB format is calculated with two vectors. A backwards vector references a macro block-sized region in the previous I or P VOB frame, the forward vector references macro block-sized region in the next I or P VOB frame. For this reason, I and P VOB frames are placed in the coded stream before any B frames that references them.

The macro block-sized regions referenced by the motion compensation vectors are averaged to produce the motion estimate for the macro block being decoded. As with P VOB frames, the error between the prediction and the frame being encoded is com-

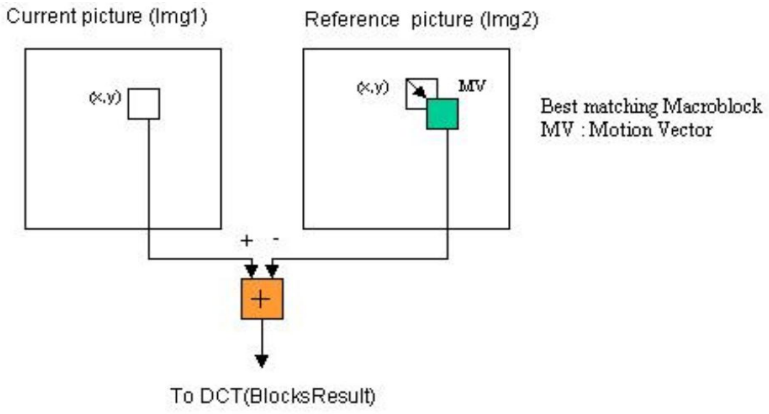


Figure 2.4: MPEG Motion Compensation Coding P Block

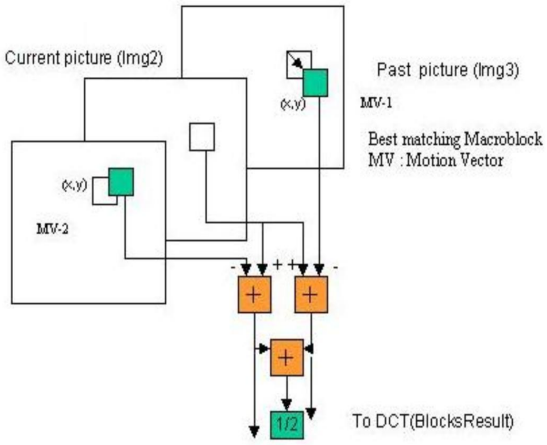


Figure 2.5: MPEG Motion Compensation Coding B Block

pressed and placed in the bitstream. The error factor is decompressed and added to the prediction to form the B frame macro block.

Many demanding techniques issues are raised by the MPEG specification. These include fast algorithms for the DCT, fast algorithms for motion vector estimation (more information is shown in chapter3, when we provide more details of these algorithms in section3.4.1), algorithms for adaptive quantization, and decompression in environments that allow some errors.

Figures 2.4 and 2.5 are examples of MPEG picture type sequence and their dependencies. P picture depend on the previous reference frame (I or P), while B pictures depend on the previous and the next reference frames.

A predictive-coded MB (Fig. 2.5) consist of a motion vector $\overrightarrow{f}$ and an error image block E, so that the content of the MB can be reconstructed using the contents of the

reference picture:

$$I(x,y)= I_{fwd}(x+f_x,y+f_y) \tag{2.2}$$

Where *I(x,y)* is the 8x8 block of image I whose top-left corner is located at (x,y), and I_{fwd} is the forward reference image. In the case of bidirectional predictive coding, there are two motion vectors $\overrightarrow{f}$ and $\overrightarrow{b}$ the MB contents are reconstructed as a function of the forward and backward reference picture.

$$I(x,y) = \frac{I_{fwd}(x+f_x, y+f_y) + I_{bwd}(x+b_x, y+b_y)}{2} + E \tag{2.3}$$

Where I_{bwd} is the backward reference picture. The resolution of a MV can go up to half a pixel or up to a quarter of a pixel in MPEG-4. In these cases, the contents are linearly interpolated from the corresponding reference picture.

2.6 Discrete Cosinus Transform (DCT) Behaviour

Moving continuous tone images are represented as a sequence of "frames". A frame is a two-dimensional array of pixel values in one "plane" for black and white images, or more planes for color images. A possible model of the signal being sampled (a sequence of pixel values forming a row, column, or time-varying sequence) as a random variable with a mean of zero. The probability distribution of pixel x_1 given the value of pixel x_0 has been shown empirically to be an exponential (laplacian distribution)[110]

$$P(X = x1 - x0) = \frac{e^{-\lambda|x_1 - x_0|}}{2\lambda} \tag{2.4}$$

Intuitively, this means that if pixel x_0 is red, there is a great likelihood that pixel x1 is red. This motion is expressed with the probabilistic notion of covariance. N samples of the signal are considered, forming the sample vector x. Exponential distribution form a stationary process, where the random function may be described by its auto-covariance matrix A (in Equation: 2.5), where:

$$\mathbf{A} = \mu(\mathbf{x} \cdot \mathbf{x})^T \tag{2.5}$$

For an exponential distribution :

$$A_{i,j} = \rho^{|i-j|} \tag{2.6}$$

where $0 \leq r \leq 1$ is the correlation between samples (r is a function of 1).

The Discrete Cosinus Transform transforms the basis set in the sample space into a new basis set such that the greatest energy is contained in the fewest number of transforms coordinates, and the total representation entropy of the sequence of coefficients in transform space is minimized. In other words, the greatest energy is contained in the earliest coefficients in the basis of the transform space.

Formally, it is said that the DCT Transform should minimize the Mean Square Error in any truncated representation of the samples in the transform space, Given a random vector $\mathbf{x}$= $(p_0, p_1, ..., p_{N-1})$, we want to find a set of basic vectors $\beta = (\beta_0, \beta_{1,...,}\beta_{N-1})$,so we can rewrite x in this basis as $x_k = (k_0, k_1, ..., k_{N-1})$. The β is chosen in a way that a truncated representation $\mathbf{t}$ of $\mathbf{x}$, given as $t_k = (k_0, k_1, ..., k_{M-1}, 0, 0, ...0)$ has minimum error:

$$t = \sum_{i=0,M-1} x_i \beta_i \tag{2.7}$$

It can be shown that this transformation in [110]:

Problem 4. Completely decorrelates the signal in transform space,
• Minimizes the mean square error in its truncated representation (guarantees minimum error in compression),
• Concentrates the most energy in the fewest number coefficients,
•Minimizes the total representation entropy of the sequence.

For highly correlated images (t approaches 1), the basis vector are exactly the Forward Discrete Transform (FDCT) [111]:

$$S_u = \sum_{i=0,N-1} C_u \cdot \cos(\frac{(2 * i + 1) * u * \pi}{2 * N}) \tag{2.8}$$

where:
$C_u = \{\sqrt{\frac{1}{N}}, u = 0$
$\sqrt{\frac{2}{N}}, u > 0$
In the DCT space the energy is concentrated in few coefficients. The energy distribution in pixel space energy is equally distributed over all pixels, but in DCT space energy is concentrated in few coefficients [109].

2.6.1 Reconstruction of Missing Compressed Data: DCT coefficients

The forward and inverse DCT of MxN = 8x8 block is defined as follows:

$$c(i,j) = \frac{1}{4} k(i) k(j) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \cos \frac{(2m+1)i\pi}{16} \cos \frac{(2n+1)j\pi}{16} \tag{2.9}$$

$$f(i,j) = \frac{1}{4}\sum_{m=0}^{M-1}\sum k(m)k(n)c(m,n)\cos\frac{(2i+1)m\pi}{16}\cos\frac{(2j+1)n\pi}{16} \tag{2.10}$$

for all *i, j = 0,1, ... ,7,* where

k(i)= $\{\frac{1}{\sqrt{2}}i$ if(i =0)

1 otherwise

This linear transform can be expressed in matrix form:

$$DCT(I) = AIA^T \tag{2.11}$$

Where the elements of A are:

$$a(i,j) = \frac{k(i)}{2}cos\frac{(2j+1)i\pi}{16} \tag{2.12}$$

The lowest order coefficient of the DCT (known as DC) defines the energy level of the original function and, in practice, is given by 8 times the average value in the block, as derived from equation 2.9:

$$c(0,0) = \frac{1}{8}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n) \tag{2.13}$$

2.6.1.1 Fast One-Dimensional DCT Techniques

Fast algorithms have been proposed for the one-dimensional Discrete Cosine Transform. They are based on the following techniques [112, 110, 95]:

a) Exploiting Symmetries of the Cosine Function
b) Relating the DCT to the Discrete Fourier Transform (DFT)
c) Relating the DCT to other discrete transformations
d) Computing using matrix representation
e) Reducing the DCT to polynomial evaluation

The first two techniques based on exploiting symmetries of the cosine transform and relating the DCT to DFT, are the fastest known techniques.

2.6.1.2 Two-Dimensional DCT algorithms

Two dimensional DCT algorithms, proposed in the literature [112, 110].
include:
a) Reducing to 1-D DCT
b) Relating the 2-D DCT to 2-D DFT
c) Relating the 2-D DCT to other 2-D discrete transformations

d) Block Matrix Decomposition

We have used in this thesis the a) 1-Dimension DCT implementation because it is a way to implement two times the 1-dim DCT. This allows the use of different Functional Units (FU) computing in parallel. As we will explain more thoroughly in chapter 4 section 4.2and chapter 5 section 5.1.3.

2.6.1.3 Three-Dimensional DCT algorithms

Compression strategies are chosen based upon the statistical behavior of the data to be compressed. Continuous tone picture have traditionally been modeled as a stochastic sequence of random variables. An autoregressive model is developed by using a casual minimum variance representation for the stochastic sequence. It is expressed the value of pixel x_n as the sum of a casual prediction x_{pn} and the error term e_n :

$$Sx_n = x_{pn} + \varepsilon_n \tag{2.14}$$

The prediction term is chosen to minimize the error term e_n. The minimum occurs when x_{pn} is the conditional mean:

$$x_{pn} = E(x_n|x_{n-1}), i = 1, 2, ...n \tag{2.15}$$

If the x_n process is Gaussian, the predictor is linear. It is assumed a first-order autoregressive model, where r is the one-step correlation coefficient:

$$x_{pn} = \rho \cdot x_{n-1} + \varepsilon_n \tag{2.16}$$

The variance s^2 of e_n is:

$$\sigma^2 = \sigma_x^2 \cdot (1 - \rho^2) \tag{2.17}$$

Lloyd and Max have developed the optimal quantizers by minimizing the mean square error of the introduced quantizing noise [113, 114]. In this case, the optimal quantizers are determined using the variance of the stochastic variables in the transform space to predict the error introduced in pixel space.

2.6.2 Defining an Invariant Measure of Error

The problem of adaptive quantization requires prediction of the error caused in pixel space by the introduction of error in DCT space. This problem is addressed by recalling that the DCT (and all unitary transformations) are distance preserving (the norm of the sum/difference of two vectors is invariant, Parseval's relation):

$$\sum_{x=0,n-1} (s_1[x] - s_2[x])^2 = \sum_{u=0,n-1} (S_1[u] - S_2[u])^2 \tag{2.18}$$

where:

s_1 ,s_2 are expressions of pixel values in pixel space,

S_1 ,S_2 are expressions of pixel values in *frequency* space, and the DCT is a unitary transform (i.e. $\mathrm{DCT}^{-1} = \mathrm{DCT}^T$)

The Mean Square Error (MSE) is defined as:

$$MSE = \sum_{x=0,n-1} (s[x] - s_q[x])^2 \tag{2.19}$$

where:

s is the pixel value in pixel space, and

s_q is the pixel value in pixel space after quantization.

Thus Mean Square Error is invariant under the DCT transformation. We define the invariant measure of error Normalized Root Mean Square Error (NRMSE) as:

$$NRMSE = \frac{\sqrt{\frac{1}{N} \sum_{x=0,n-1} (s[x] - s_q[x])^2}}{\mu} \tag{2.20}$$

where μ is the pixel value (in pixel space).

The foundation can now been laid for the definition of a criterion for measuring used to describe the discrepancy introduced when a component of the DCT is recovered from its quantized value. Quantization is defined as the rounded quotient of the DCT component and a quantizing factor:

$$Q(x) = [\frac{x}{q} + \frac{1}{2}] \tag{2.21}$$

where:

x is the DCT component to be quantized,

q is the quantizing factor, and

Q(x) is the quantized value of x.

Definition 5. Quantizing error is defined as:

$$x' = q \cdot Q(x) \tag{2.22}$$

$$E_q = x - x' \tag{2.23}$$

where x' is the dequantized value of x. is E_q

2.6.3 Adding Human Visual Factors

It may be desirable to modify the optimum quantizers by weighing them with factors determined by evaluating human visual acuity. By analogy, MPEG continuous tone picture quantizers are skewed by factors intending to take advantage of the reduced human visual sensitivity to rapid small changes in the picture. It is well known that the human eye is insensitive to motion changes of $\frac{1}{60}$ of a second, and this fact was used to establish the frame rate of television transmission.

The process of quantizing coefficients in DCT space relies heavily on the decorrelation property of the DCT. Since the DCT decorrelates the random variables in DCT space, each DCT coefficient may be individually processed. Now a different view of DCT coefficients can be developed.

A model of human visual acuity can be developed based on DCT coefficients. Since DCT coefficients can be individually processed, it is also possible to study their individual visibility. Two dimensional human visual acuity have been modeled by a modulation transfer function of radial frequency in cycles/degrees of visual angle subtended. The results of this work have been published in [115] .The relative visibility of DCT basis vector in the spatial dimensions is documented in table 2.5

362	362	362	362	362	362	362	362
502	426	284	100	-100	-284	-425	-502
473	197	-196	-473	-473	-196	196	473
426	-100	-502	-284	284	502	100	-425
362	-361	-362	362	-361	-361	-362	362
284	-502	100	426	-425	-100	502	-284
197	-473	473	-196	-196	473	-473	196
100	-284	426	-502	502	-425	284	-100

Table 2.5: Relative visibility of DCT basis vector in spatial dimensions.

2.7 Entropy Coding

Binary encoding of data is a natural means of representing computational data on modern digital computers. When the values to be encoded are uniformly distributed, this is an space-efficient means of representing the data as well. Information theory gives us several efficient methods of encoding "alphabets" where the likelihood of symbol occurrence varies symbol by symbol. Coding techniques that minimize space in the representation of random sequences of symbols (they optimize space used in the representation of symbols based upon the probability the symbol appears) are known as entropy coding techniques.

There are two popular methods of entropy coding in the literature, Huffman coding and arithmetic coding. Huffman coding represents symbol with words of integer length

while arithmetic coding is not limited to integer-length codes. Huffman coding is computationally less expensive to implement and typically gives compression ratios close to those of arithmetic coding.

2.7.1 Huffman Coding

Consider the problem of encoding the six symbols defined in table *2.6*. The amount of information transferred in a symbol A that occurs with probability p is:

$$I_A = \log_2(\frac{1}{P_A}) \tag{2.24}$$

where I_A is the number of bits required to express the amount of information conveyed by symbol A, and P_A is the probability that symbol A will occur, The entropy of a code sequence is the average amount of information contained in each symbol of the sequence:

$$H = \sum_s p(s) \cdot \log_2(\frac{1}{p(s)}) \tag{2.25}$$

where H is the entropy of the coding representation, and s ranges through all symbols in the alphabet of symbols.

Symbol	Probability	Information	Code
A	1/2	1 bit	0
B	1/4	2 bits	10
C	1/16	4 bits	1100
D	1/16	4 bits	1101
E	1/16	4 bits	1110
F	1/16	4 bits	1111

Table 2.6: Symbols and their associated Huffman code

The entropy of the sequence represents the lower bound of the space needed to communicate the information contained in the sequence. A fixed word length of three bits may be used to represent the six symbols in table 2.6. Using the Huffman coding representation, it is got an average code length of 2, which for these probabilities happens also to be the entropy, or lower limit of the average code length:

$$H = (\frac{1}{2})x_1 \ + \ (\frac{1}{2})x_2 + (\frac{1}{16})x_3 + (\frac{1}{16})x_4 + (\frac{1}{16})x_5 + (\frac{1}{16})x_6 = 2 \tag{2.26}$$

Assignment of Huffman codes is done by developing a Huffman coding tree. The tree is developed "left to right"(or bottom to top). Symbols are listed in decreasing order of probability. Iteratively, the two project branches whose sum is least are combined into a new branch, until the tree is completed. The tree is then traversed and labeled.

This algorithm computes Huffman codes for arbitrary sequences of symbols, and will work regardless of their relative probabilities (real cases do not show the simple logarithmic behavior of this example). The algorithm is simple enough (O(NlogN)) to form the kernel of a real-time algorithm for the adaptive entropy coding of images. Huffman coding is used in the JPEG and MPEG specification.

2.7.2 Arithmetic Coding

Entropy coding schema based on codewords that are an integral number of bits long (such as Huffman coding) cannot achieve optimal compression of every set of data. This is because the theoretical optimum number of bits is the "information content" a fraction (rather than an integer). This optimum number of bits is the "information content" $\log_2(\frac{1}{P})$, where P is the probability of occurrence of each data symbol. Arithmetic coding provides a practical alternative to Huffman coding and can more closely approach the theoretical maximum compression [116]. An arithmetic encoder converts a sequence of data symbols into a single fractional number. The longer the sequence of symbol, the greater the precision required to represent the fractional number.

A number of patents have been filed that cover aspects of arithmetic encoding. It is not entirely clear whether the arithmetic coding algorithm specified in the image and video coding standard are covered by patents. Some developers of commercial video coding system have avoided the use of arithmetic coding because of concerns about potential patent infringements, despite its potential compression advantages.

2.8 Summary

This chapter explains how human eye works and its limitations to distinguish color, number of frames per second, or the loss of sensitivity for high frequencies. MPEG encoder profits these facts to compress sequences of images and remove superfluous information that the human eye processes, without that the eye might realize the differences between original image sequence and the compressed one. It is clear that if the compression ratio is too high, the human eye is going to perceive that the sequence is loosing quality; therefore, the PSNR concept is introduced to calculate this loss Chapter introduces briefly the different compression MPEG standards from the oldest to the current ones (MPEG-1, 2, 3, 4, 7 and 21) and several references are provided to get more information to go in deep in the MPEG world. Furthermore, details are given about the MPEG encoder that may choose compression options balancing the need for high compression ratios against the complexity of motion compensation or adaptive quantization calculations and all the concepts that derive from this idea such as: Need of real time, requirements of insensitivity to error transmission, and fast algorithms consequently use of specific hardware. Finally, basic algorithms for spatial and temporal compression are also described in a mathematical way. These are used in MPEG for VO compression, the MC and ME (temporal) and DCT(spatial) to explore the cost-performance trade-off at the system level.

3 Application Implementation in Multi-Platform for Energy-Flexibility space Exploration

Today, embedded multimedia devices are widely present in our lives. Choosing the right main computing element is a key issue for the success of these devices. Consumption, performance, retargetability and development time are some of the elements that need to be analyzed and well balanced. In this chapter, we present the mapping of the same multimedia application (MPEG-4 Main Profile) into different target platforms typically used in the embedded domain. The design flow of this work begins on a MPEG-4 encoder described in Matlab and translated to C++ language which is converted to a SystemC hardware/software description. Afterward, the code is refined and optimized for the execution on multiple platforms: an Application Specific Instruction Processor (ASIP), a soft-core processor with specific functional units implemented on an FPGA, an Application Specific Integrated Circuit (ASIC) and an embedded platform on a single chip formed by a high performance DSP and a processor.

.

3.1 Introduction

Nowadays, embedded multimedia applications are widely present in our lives. Those applications are usually specified in System-Level Design Languages (like Java, UML, C++), starting from a reference or golden model, that needs, in many cases, to be executed in real-time cost/energy-sensitive devices as mp3-mp4 players, mobile phones, personal data assistants (PDAs), etc. Those devices have to be very performing, because of the applications requirements, at low energy consumption due to battery life. Choosing the right main processing element is a key issue for the success of those devices where consumption, performance, retargetability and developing time are some of the elements that need to be analyzed and balanced. Actual designs can include different processors: application specific integrated circuits or instruction set processors (ASICs or ASIPs), general purpose processors, various digital signal/media/image processors, embedded FPGA, etc. with their own local memories, architecture and with a complex interconnection scheme[117]. Attending at the increasing computational complexity of multimedia applications present and future System on Chip (SoC) platforms have to satisfy many requirements: high computation, enormous quantity of memory accesses and high communication [118, 119] flow between different processing elements for non-deterministic applications. Moreover, embedded multimedia applications must satisfy

hard real-time constraints maintaining an acceptable quality of service for the users at the lowest energy consumption. To satisfy these computing requirements, most present SoCs will contain several types of processor cores and memory units ([120, 121]) connected through a hierarchical shared bus, point to point, bus matrix or a NoC[25]. These complex platforms can handle real-time video and audio compression and are usually based on heterogeneous solutions containing at least one DSP for multimedia-data-flow acceleration and one processor core for control flow and reactivity. In this article, we propose a comparison of different implementations coming from an original unique golden model of the application. Current comparisons found in literature [119, 122] are based in small kernels or not fully platform-optimize applications. Without real aggressive optimizations, tuned for each individual style, the comparison of the results typically give an unfair advantage to one or other platform. As far as we know, our work is the first to compare different design options for modern multimedia applications, coming from a unique, realistic and complete application: namely a MPEG-4 Video Main Profile (MP) specification. In our work, we mapped the same application into different platform styles, applying strong platform independent and dependent optimizations. Comparing optimized implementations from a single unique real source gives realistic and significant results since we are making an impartial comparison.

The main purpose of our work is to quantitatively recognize the fundamentals that system designer requires for allowing a priority and very early in the design path, which platform method/subclass is the most convenient choice to implement the target system. The outcomes are exposed for a video compressor solution but they can be extended to any data-flow dominated system. Most multimedia applications at the present time fall into that category. In addition in this manuscript we make available current comparative values among the different platforms and the results come from a single original description with sufficient effort spent on each of them to come to a sufficiently optimized solution. As a result, these choices are evaluated also in an objective way. Furthermore, this chapter studies general optimizations for the embedded system domain and the links between the different implementation styles and the final targeted platforms and related optimizations. The remainder of the chapter is organized as follows. In Section 3.2, we describe the state of the art, where we explain the different MPEG platform architectures and the differences in terms of energy that we can obtain depending on the target platform. Section 3.3 details the models and platform description of the overall system and the different algorithms that compose the MPEG application. Section 5.4 shows the diverse customized solutions under study and the metrics used to evaluate them. In this section, we also present the individual results of each platform and the comparison between the different solutions. Finally, in section 5.6, we summarize our work and we present our chapter conclusions.

3.2 State of the art

As already mentioned in chapter 2, MPEG-4 is a global multimedia standard that delivers professional-quality audio and video streams over a wide bandwidths range, from

cell phones to broadband, HDTV and beyond. MPEG-4 was defined by the Moving Picture Experts Group (MPEG), the working group within the International Organization for Standardization (ISO) that specified the widely adopted standard. The standard is divided in levels and profiles [123]. A survey with diverse MPEG implementations exists in reference [124]. This paper provides an overview on the MPEG-4 standard in general with a special emphasis on the implementation of the video coding part. The properties of the most popular visual profiles, their application fields, characteristic implementation and examples are discussed. In their review of current state-of-the-art and future developments in MPEG-4 circuit design known from literature includes dedicated hardware as well as programmable architectures, such as embedded RISC-based (software) solutions, DSPs, and media processors. Strictly speaking, the majority of the existing implementations belong to the group of the hybrid architectures, which are based on the combination of a programmable CPU, either a RISC or a DSP, for the control part and a number of dedicated hardware accelerators for the encoding or decoding process. The solutions provided have different number of hardware accelerators depending on the partition of the architecture and the processing capabilities of the programmable core. Embedded processing is of an increasing importance [125] and most devices target the mobile domain (videophones, PDAs, digital video cameras, etc).

A number of programmable approaches exist for the implementation of MPEG-4 visual profiles which can roughly be divided into four classes: 1) based on general-purpose processors (GPPs) with media instruction-set architecture (ISA) extensions; 2) based on embedded RISC processors; 3) based on traditional DSP architectures, and 4) based on special mostly very long instructions word (VLIW) media processor. A good overview on ISA extensions for GPPs is given in reference [126]. The main focus of MPEG-4 Visual is on compression efficiency, object-based functionalities, scalability of textures, images, shape and alpha channel coding, error-robustness, face and coding of 2D/3D meshes. MPEG-4 Visual comprises a toolbox of different coding instruments. The combination of specific coding tools is organized in profiles and levels, each targeting a preferred application field. Reference [127] provides a survey of state-of-the-art hardware architectures for image and video coding. Fundamental design issues are discussed with particular emphasis on efficient dedicated implementation. Hardware architectures for MPEG-4 video coding and JPEG 2000 still image coding are reviewed as design examples, and special approaches exploited to improve efficiency are identified. Further perspectives are also presented to address challenges on hardware architecture design for advanced image and video coding in the future according with new formats and standards.

Some of previous efforts in the area of reconfigurable computing have demonstrated that dynamic matching between application and architecture leads to spectacular energy savings for signal-processing applications while maintaining the required flexibility. System engineers must decide which parts of the application should be implemented on hardware or software, hence trading-off between energy consumption and performance versus flexibility and fast developing time, trying to reach optimum results in this complex design space. Authors show in [128] that processor under study implementation is three orders of magnitude more expensive (in terms of energy consumption) than the equivalent on dedicated hardware.

The paper [128] also claims that reconfigurable architectures which utilize a "programming-in-space" approach have proven to be very efficient for the signal-processing part of the communication protocol, delivering high-performance and enough flexibility to adapt to the varying conditions of the system and environment at an energy cost that comes close to a custom hardware implementation.

Their experiments showed that FPGA and configurable finite-state-machine implementations of a protocol stack are two orders of magnitude more efficient than embedded microprocessor or microcontroller solutions. This will be true for small and arithmetic or control-dominated examples. In our work we demonstrate that for modern multimedia applications, that typically exhibit complex signal processing loops and a large amount of memory access, a DSP or an ASIP based solution offer better results in terms of energy efficiency than fine-grain reconfigurable processor logic.

In the same publication, authors claim that more coarse-grain reconfigurable processors such as Pleiades [129] are also more energy efficient (in terms of MOPs/mW) than DSPs and ASIPs. This is again mostly because of the arithmetic/control-dominated focus. Moreover, they did not compare the same codes: their comparison was based on similar but not exactly the same application mapped into different platforms. In contrast with their work, in our work we propose to target a representative and complex code, specifically the MPEG-4 Main Profile (overall and not only a few loop kernels), which comes from exactly the same unique golden model. This original golden model is then transformed and optimized to be targeted in various platforms (ASIC, FPGA, DaVinci DSP and an ASIP) to get more realistic numbers about power efficiency and a fair evaluation of the capacities of each platform.

Today's benchmarks used in platform characterization are not designed to address any particular user problem (consumption e.g.), and hence lack of meaning and effectiveness. Instead, they are in general intended to quantify system's performance, whether it is hardware or software. Although such micro-benchmarks [130] can be useful in understanding the end-to-end behavior of a system, in isolation they are frequently uninformative about the overall system performance, primarily because that performance is highly workload-dependent [131]. As an example of this, we can consider the already mentioned paper [122]; this article reports work on a performance benchmark of different implementations of some low-level vision algorithms. These algorithms are implemented on two high-speed embedded platforms based on a Digital Signal Processor (DSP) and a Field Programmable Gate Array (FPGA). Again, the problem in the comparison of the results is that the algorithm does not come from the same model; one comes from a C description and the other from an RTL (Register Transfer Level) description. In reference [122], the DSP has better results but just takes one task into account and does not consider multiple tasks running in a concurrent way as it is done in the FPGA. Another example, shown in [132], is a coarse-grained reconfigurable image stream processor for digital still cameras and camcorders that is compared with other similar solutions based on processors or ASICs, but the results obtained come from different source codes and these solutions are targeted to different silicon process technologies. Therefore, it is complex to evaluate the real performance of the diverse resolution since they are evaluating apples with pears. Another problem that invalidates the conclusive nature of

those results, is that these algorithms are relative small when seen in the context of the multimedia domain and hence not really representative for that domain.

When benchmarks come from different sources it is reasonable to think that they come from diverse written styles although they all follow the same standard. Benchmarks might have been written by different engineers or by the same person using different patterns. An example of this fact can be found in [124] where there are twelve multimedia benchmarks among video compression, decompression and image processing models under study. These benchmarks target two different processor models, one being an order processor and the other an out-of-order processor. Unfortunately, these benchmarks have not been tuned to fit the logic and memory characteristics of the objective platforms and hence the conclusions on the performance results might guide to wrong conclusions. Furthermore, instead of modifying the benchmark, the authors recommend to change the processor architecture to better fit the benchmark. Especially for modern multi-media applications, an algorithm-architecture co-design exploration is essential, with a crucial role for optimizing source code transformations.

Some of the previous efforts in the area of reconfigurable computing have demonstrated that dynamic matching between application and architecture leads to spectacular energy savings for signal-processing applications while the required flexibility is maintained. System engineers must decide on what parts of the application go to hardware and what to software and hence, they make a trade-off between energy consumption and performance versus flexibility and fast developing time, attempting to reach optimum results in this complex design space. In [128], the authors show that the processor implementation is three orders of magnitude more expensive (in terms of energy consumption) than the equivalent in dedicated hardware. Paper [128] claims that reconfigurable architectures, which use a "programming-in-space" approach, have proven to be very efficient for the signal-processing part of the communication protocol, delivering high-performance and enough flexibility to adapt to the varying conditions of the system and environment at an energy cost that gets closer to a custom hardware implementation. Their experiments showed that FPGA and configurable finite-state-machine implementations of a protocol stack are two orders of magnitude more efficient than embedded microprocessor or microcontroller solutions. This is true for small and arithmetic or control-dominated examples. In our article we demonstrate that for modern multimedia applications, those generally have complex signal processing loops and a large amount of memory access; a DSP or an ASIP based solution offers better results in terms of energy efficiency than fine-grain reconfigurable processor logic. In the same publication, the authors claim that more coarse-grain reconfigurable processors such as Pleiades [133] are also more energy efficient (in terms of MOPs/mW) than DSPs and ASIPs. This is again mostly because of the arithmetic/control-dominated focus. Moreover, they did not compare the same codes: their comparison was based on similar, but not exactly the same, applications mapped onto the different platforms. In this article, in contrast with their work, we propose developing a representative, complex code, namely the MPEG-4 Main Profile (overall and not only a few loop kernels), which comes from exactly the same unique golden model. This original golden model is then transformed and optimized to be targeted in various platforms (ASIC, FPGA, DSP and ASIP) in order to obtain more

realistic values of power efficiency and a fair evaluation of the capacities of each platform. In a previous work [134], we detailed results for different platform MPEG implementations. We made a fair comparison between platforms but we did not provide an accurate explanation about the path followed to obtain the results. In this thesis, we show the different platform-independent and platform-dependent optimization steps for obtaining optimal results in the different platforms and an improved percentage between a raw and an optimal implementation. As we will see in Section 5.2 , an original code, coming from the golden model, directly mapped into a non-optimized ASIP needs around 30 more clock cycles, which consumes 40 times more energy than an optimized code for an optimized ASIP. This optimized ASIP can compete with the optimized ASIC, giving results of the same order of magnitude. ASIC results are better than any other platform.

3.3 Model and Platforms Description of the Overall System

The objective of early SoC-design analysis and hardware/software co-design is to obtain the right allocation of functions to hardware and software and to correctly size hardware resources to meet the design requirements. At this early stage of development, some major SoC-configuration decisions must be made which will affect the final implementation of the design, and hence the platform results. When engineers start a project they have a broad design space, and the different possible implementations have distinct advantages and drawbacks that lead to huge differences in the quality of the results. In the following sections we describe the main distinct platforms that engineers can use and the different metrics needed to evaluate their suitability for each solution. In this work we will not consider the economic aspect of the different implementations, which are well known in industry, and will just focus on the technological differences of the targeted platforms that can handle our driven application (MPEG4) efficiently.

3.3.1 Introduction to work-flow scheme

In this section, we give an overview of the complete flow followed during the work and we develop further the parts that require a more in-depth explanation. The model to be implemented in the different platforms is based on a video compressor with main profile and permit I, P and B slice compression, with 4:2:0 chroma format, and 8 bit pixel sample depth [123, 135, 136]. This model comes from an original “golden model” code developed following the MPEG standard completely, the main parts were developed from scratch and the other parts of the model were obtained from open sources already developed. The most computationally-intensive algorithms were produced using the Matlab framework [69]. These modules are the motion estimation (ME), the motion compensation (MC), discrete cosine transform (DCT), the quantization (Q), and zig-zag;. for the rest of this thesis, we will call these algorithms the kernel part of our application. These algorithms are data-flow oriented and hence very intense in terms of computation. They are an excellent potential target to be accelerated. Other algorithms involved in the MPEG compression are the entropy coder and the algorithms that mount the MPEG video stream; these algorithms are more control-flow oriented and thus harder to accelerate.

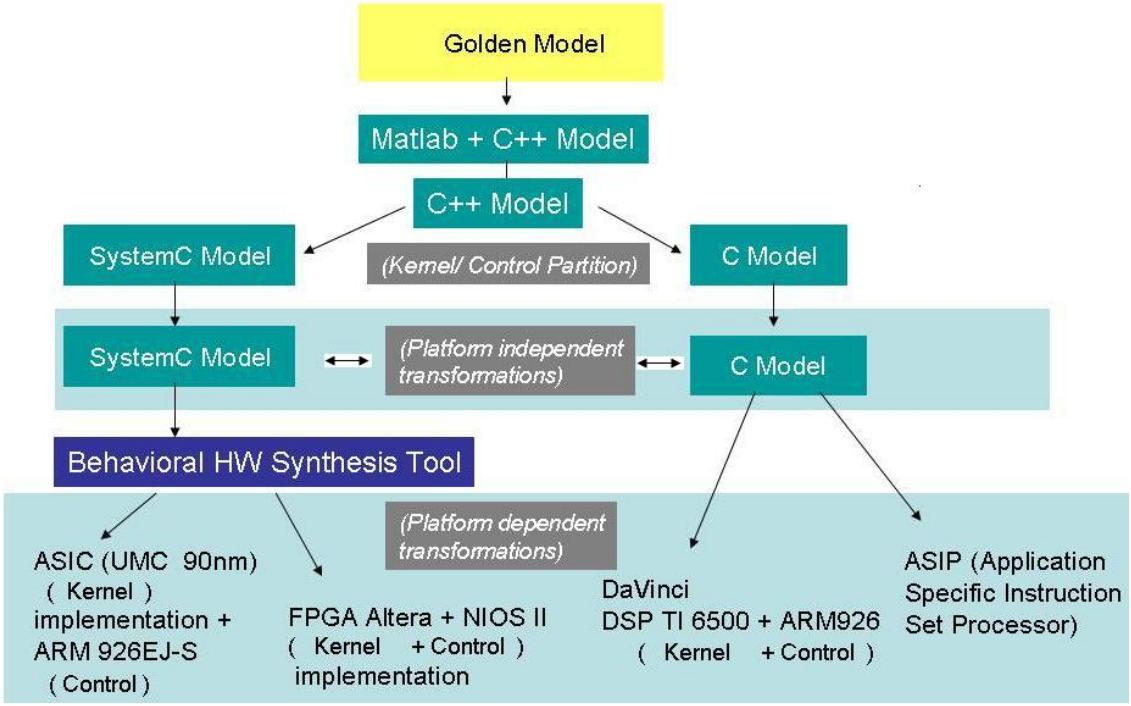


Figure 3.1: MPEG design flow from concept to platform

We used Matlab to develop the application kernel since it provides an excellent framework for data and signal processing. Once this model was finished, we used it as a reference code for coherence and subsequent verification when we developed a C++ version of the model. At this abstraction level, some platform independent optimizations where developed and the refined C++ model was used as a transition step to two new versions: a SystemC [81] version and a C version. These two new versions will be used later on to map the application on an ASIC and a FPGA (SystemC version) and a commercial DSP and an ASIP (C version)[137]. Figure 3.1 shows a diagram of the path followed.

As mentioned above, the model has two differentiated parts: the control flow part and the kernel part. When the ASIC and the FPGA are targeted, the SystemC model related to the kernel is passed through a behavioral hardware synthesis tool which creates a low level RTL description in a hardware description language. This description can be used later for synthesis in the FPGA or the ASIC, and will serve for performance estimations of the hardware parts, although it is not really optimized. The control part of the application does not need the power of a hardware implementation and a simple RISC processor (an ARM9 in the case of the ASIC or a NIOS II in the case of the FPGA) can handle this part of the algorithm perfectly well; as it is more control-flow oriented, a software implementation is more straight forward. Before both models (SystemC and C) are mapped onto the final platforms, they were exposed to some platform-independent and platform-dependent optimizations to obtain the maximum benefit for each platform

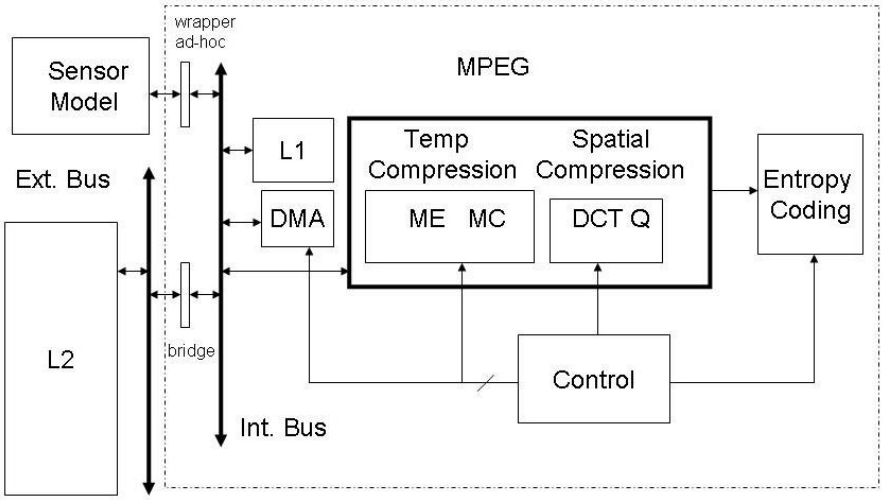


Figure 3.2: MPEG Model Diagram

style. These optimizations will be analyzed in detail in Section*3.5.*

3.4 Work-flow scheme: C++ Model

The model envisioned is a MPEG compressor that completely follows the standard. We have also fulfilled a sensor model that acquires images and saves them in a level two (L2) memory. A DMA module then fetches these data pixels from the L2 and transfers them to a level one (L1) memory. This module divides the entire image into the blocks that have to be compressed. There is an ad-hoc wrapper that connects the sensor with the internal bus. There is also a bridge that connects the external bus and memories with the internal bus. Figure 3.2 illustrates a block-diagram of the MPEG model in which the dashed line represents the functionality of all the MPEG compression modules. Blocks inside the bold square represent computing intensive algorithms (the kernel) and are the part of the application that will be optimized. The kernel algorithms can be separated into temporal compression and spatial compression. The motion estimation (ME) and the motion compensation (MC) are the main algorithms for the temporal compression part of the kernel and discrete cosine transform (DCT), quantization (Q) and entropy coding are the main algorithms for the spatial compression part of the kernel. These algorithms are explained more deeply in the following sections. The MPEG kernel was originally developed in Matlab then translated into a "special" C++ model. This "special" C++ model does not have dynamic memory allocation, global variables or pointers because the model was intended to be implemented in both software and hardware, which has languages and tools with requirements that are not necessary for software implementations.

3.4.1 Functionality: MPEG System

The MPEG standard is based on two main types of compression: temporal and spatial. The main algorithm related to temporal compression (compression between adjacent images is the motion estimation and motion compensation) is the ME. In this work, we develop two different ME implementations, a FSME (Full Search Motion Estimation) and a FAST solution; First, all MV (Motion Vectors) candidates are computed to achieve the minimum SAD (Sum of Absolute Values) [135]. Second, the faster solution is a logarithmic approach we obtain a local minimum with [138]. The algorithm related to spatial compression is the DCT (discrete cosine transform) [123, 135, 139]. The frequencies obtained in the output of this algorithm have to be quantized, and then the entropy result must be encoded with the arithmetic compression. In the following section, we will briefly explain different aspects of the overall multimedia platform description model used in our work.

3.4.2 Sensor Model and Direct Memory Access

The modelled sensor that is used for image acquisition is a generic CMOS [140] image sensor. This sensor includes a 2592x1944 image array and a 10-bit A/D converter capable of operating at 7.5 frames per second (fps) in full resolution; the chip set works at 96 Mhz. The sensor can also operate at 60fps at 864x648 resolution enhanced video viewing on TV. This sensor supplies a YUV (Y Luminance represents brightness and UV Chrominance represents colour information) channel output. Therefore, we do not have to transform the 8 bit RGB (red, green, blue) channels to YUV because the conversion is carried out directly in the sensor and produces pixels in three YUV channels. The YUV output format used for our experiments is 4:2:0 because human eyes are more sensitive to brightness than colour; therefore, the Y channel has four times more information than the U or V channels. The acquired images are saved in a L2 memory. The optimal size of this L2 memory depends on the number and size of the P and B frames. Remember that to compute a P frame the reference frame is required (which must be stored in the memory), and to compress a B frame the reference and the future frames are needed.

The Direct Memory Access (DMA) module acquires the data from the external memory (L2), where images are fetched from the internal memory (L1) in macro-block structures. The data structures fetched depend on the image type that the MPEG is encoding. If the image to be encoded is type I (only spatial compression is necessary, similar to JPEG, and therefore based on DCTs) the data structures fetched to L1 are four blocks of 8x8 pixels for channel Y and 2 blocks of 8x8 pixels for channels Y and V in a 4:2:0 compression. If the macro block to be compressed is a P type (temporal compression between a reference image and the current image) the residual is later spatially compressed, then the structure fetched is similar to that of image type I, but in this case, it is also necessary to send a pixel structure p by p in size (where p is the search region that extends from 16 to 64 pixels). This structure provides the data-pixels to compute motion vectors from the reference image to the current one. Then, current and referenced image macro block data-pixels are sent to the motion compensation module. The outcome

is the residual transformed by the DCT, which is then quantized and compressed with the entropy coding algorithm. In the case of a B macro-block, the data structure to be fetched is similar to the P frame but instead of a p by p structure, two structures of this size are necessary. These structures serve to carry out the ME of the reference image and the current one, and the ME of the current and the future image. These two vectors are used to compensate images in the temporal compression and then to compute the spatial compression.

3.4.3 Platform Controller

The platform controller module is in charge of managing, at macro-block level, the kind of image that is going to be compressed. This module chooses the subsequent macro-block to be compressed, either an I, P or B macro-block, by sending the appropriate information to the DMA module. The DMA will then fetch the required data-pixel structures from the L2 memory. At the same time, the control of the MPEG module decides on the corresponding algorithms involved in the compression. Therefore, this module also controls the type of Group of Pictures (GOP) that is going to be compressed. The platform controller is also responsible for controlling the computation of the different algorithms. For this thesis, the computation in each algorithm is the maximum possible and we always consider the worst case scenario: every algorithm computes all possible data to obtain the best compromise in terms of compression/quality. Nevertheless, this module can change the computation of the whole MPEG application to adjust the compromise compression/quality and tune different parameters of the different algorithms.

3.5 Accurated Calibrated Exploration and Target Platform Comparison

In our work, we have chosen four platforms that are widely used in the embedded domain. Any of these four platforms can potentially fulfill the requirements of our code with the required optimizations, but all have advantages and drawbacks. In this section, we will first analyze the parameters needed to evaluate the different platforms and then study each of the platforms. We have two different types of costs: implementation costs and design costs. Implementation costs include area and energy consumption. The area of a chip determines the cost of a VLSI process. Contributions to this cost are due to the architecture of the system, memories, etc. Energy consumption depends on several parameters: technology, area, frequency, supply voltage, bit activity, algorithm, leakage, etc. The area of a design used to be the largest design cost, but nowadays, with the continually increasing integration capacity, the design area is not crucial, although it has to be taken into account (it is directly related to the circuit cost), and the energy consumption is now the limiting factor that drives design decisions, especially in the embedded domain due to battery life. Design costs are related to the development time and effort (in man-power and tool costs) of the system: programmable solutions have the advantage of easier software developments over the slow hardware development of

other solutions. Easy compilation and a retargetable framework are advantages that can help to boost the time-to-market of the products or can facilitate upgrades or new versions of the applications.

3.5.1 Platform Independent optimizations

In all models we carried out a Data Transfer and Storage Exploration (DTSE)[141]. The DTSE methodology that we have applied is the following: first the code has been programmed without any pointers and without using dynamic memory allocation. We have adapted the types of information to the values of these, so that for example for values that are not exceeding the value 255, we use only 8-bit registers. We have changed global loops (image size for example) to calculations to macro-block level to diminish the L1 of all the platforms, and helping the possible pipeline at block level. These optimizations are equal for the C version and the SystemC (platform independent). Transformation to reduce the size of the data structures inside the L1 memories means the memory inside the different platforms has to be small enough to fit on the chip. In addition, computation is carried out at macro block level not at image level.

In the following paragraph, we explain our FPGA and ASIC custom platform solutions based in a behavioral SystemC code synthesis. We developed an architectural exploration in relation to functional units during the synthesis process. Then, we obtained different compromises in terms of area versus execution time of the different algorithms involved in the video compression. These syntheses provide hardware code at RTL level that can be synthesized in an FPGA or in an ASIC.

3.5.2 Summary

In the chapter, there is an introduction to the system under study. State-of-the-art is presented, where existing different platforms that can implement the compressor are explained. Hence, it makes clear that till now there is not a comparative of a complex IP where the different platforms hardware trade-offs could be observed. We have made an overall description of our model and different platforms. We explained the work-flow scheme and C++ model where there is an explanation about the MPEG functionality, sensor model and memory access. We have also explained the platform controller that is the responsible to get the pixels from L2 in macro-blocks size to L1. Furthermore, it is responsible to decide what type of frame is going to be compressed. We give details that we have made an accurate calibrated exploration and target comparison. Finally, we explain the platform independent transformation that we have achieved to the code.

4 Customized Platforms Architectures

In the following chapter, we explain the solutions based on customized platforms: the FPGA and the ASIC. Those solutions are based in a behavioral SystemC code synthesis. We perform an architectural exploration at functional units level during the synthesis process. Then, we obtain different compromises in terms of area versus execution time of the different algorithms involved in the video compression. This synthesis provides hardware code at RTL level that can be synthesized in a FPGA or in an ASIC.

The main difference between SystemC[81] and a C description of the application is basically the communication between modules. SystemC modules must be synchronized with a handshaking protocol since simulation is event driven and when a behavioral synthesis is realized to produce the RTL code, the number of cycles required is unknown. Handshaking is not necessary in the C model. Our C model is very similar to our SystemC model since synthesisable SystemC code does have neither global variables or dynamic memory allocation or pointers. In our model, in addition with the communication scheme, another difference comes from the divider operator. Embedded software processors have specialized clusters or compiler aids to compute the division operation but a raw hardware divisor is too costly or the libraries needed for the synthesis might not have divisors. Therefore, we have implemented in SystemC a SRT divider [142] since all divisions are computed with integer's values.

4.1 Behavioral Synthesis Tool: Set-up

The tool used for hardware synthesis is Cynthesizer 5.4 from ForteDS[143] and generates production-quality RTL using a high-level SystemC-TLM design description as input. The main project script has information about the technology libraries; in our case we used Faraday UMC 90nm [144] and the worst case technology libraries. We have also added wrappers to the memories, these memory wrappers are completed with the software provided by Cynthesizer (*bdw_memgen*). These wrappers must have exactly the same pinout as the real memories generated with the tool that provides the technology (UMC). At this level, we have realized a study where we have synthesized the kernel with different Functional Units (FU) number and we had a trade-off between an implementation with minimal area (optimized for area), and an implementation with maximum speed (optimized for speed). The synthesis procedure is made in two steps:

4.2 Customized Platform Optimizations

The first step in the synthesis is to carry out a technology-aware architectural exploration of each part of the different algorithm involved in the MPEG application. As multimedia applications are based on very intensive loops, unrolling the inner loops of the algorithm makes visible the available parallelism and maximizes the usage of the resources. We implemented a version of the code with unrolled loops using the Cynthesizer primitive(*CYN_UNROLL*). When a loop is unrolled, several things occur and they affect the kind of hardware produced. First, Cynthesizer replicates the operators inside the loop by the number of iterations in the loop's upper and lower limits. Cynthesizer also creates a controller that regulates any iteration-specific items in the replicated hardware. Finally, we have to add all the information about the synthesis and simulation configurations. Simulation configurations can be behavioral, register transfer level with C++ code, or register transfer level with Verilog code. Then, we can include different synthesis configurations that are BASIC, FU2, FU4 and FU8 running in parallel. An additional parameter provides the number of loop unrolling in the inner most loop that will be used to create the synthesis. For example, if we take the code one dimension DCT from algorithm 4.1 , after line 4 we add a Forte directive line (*CYN_UNROLL*).

```
DCT(inputs tmp_dct1, tmp_dct outputs matriz_in_quant )
1 for ( j=0; j<M ; j++){
2       for ( i=0; i<N; i++){
3       //Cynthesizer primitive
4       CYN_UNROLL(AGGRESSIVE,(FU_dct),"unroll_inner_loop_dct");
5       tmp_dct1[j][i] = 0;
6       for ( index=0; index <M; index++){
7               tmp_dct1[j][i] =tmp_dct1[j][i]
                                + dct[j][index]*tmp_dct[index][i];
8       }//index
9       matriz_in_quant[j][i]= tmp_dct1[j][i]>>10;
10      }//i
11}//j
```

Figure 4.1: One-Dimensional DCT Code

This line unrolls the loop the number of times indicated by the variable FU_dct, and can be either 1, 2 or 4 times. This unrolling permits a trade-off between area and execution time. It has to be taken into account that the memory bandwidth must be appropriate for computing the data set concurrently. We have carried out loop unrolling of 1, 2 and 4 FU inside the inner loop, for the DCT, MC and Q algorithms. For ME the unrolling was set to 2, 4, and 8. The results can be seen in Figures 4.2a and b. In 4.2 , a) shows a bar graph with the area achievement of different algorithms involved in the MPEG application. The following information is given for each algorithm and implementation value: logic area, estimated mux area, memory area and register area.

The total size of each bar represents the total area of the algorithm with the studied loop unrolling implementation. In the case of the quantization algorithm, as there is

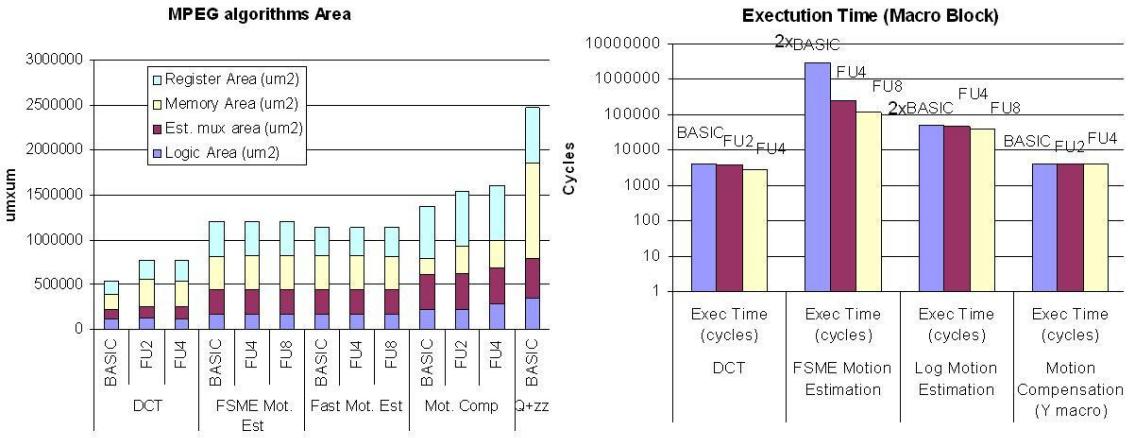


Figure 4.2: a) Area in mm^2 for implementing MPEG Algorithms (DCT, FSME ME, Fast ME and Q+zz) b) Execution time in number of clock cycles (DCT, FSME ME, Fast ME and MC).

interdependence between code structures, so, it is not possible to unroll the inner-most loop. This Quantization algorithm has the highest area because it implements the SRT division [142]. Figure 4.2 b) shows that with more FU the execution time decreases (cycles are in logarithmic scale). It also shows that the most computation-consuming algorithm by far is the FSME motion estimation and by adding more functional units (unrolling loops) it is possible to decrease the execution time. This is mainly because the SAD computation can be calculated in parallel. The fast ME is the next most computationally hungry algorithm and after that, the MC and DCT. This figure demonstrates that when the ME algorithm is the dominant algorithm, a slight increase in the area leads to a large decrease in execution time. With the MC and DCT algorithms, the decrease in execution time is not as high, but significantly enough to accelerate data-pixel treatment, with a significantly low increase in area compared with the ME algorithm.

4.2.1 HW Synthesis

Secondly, with these results, we carried out a complete synthesis, optimizing for speed but not for area. Differences appear in terms of area and speed between a BASIC (without loop unrolling) synthesis and the one optimized for speed (unrolling the inner-most loop of the ME 8 times and the DCT and MC twice). In Figure 4.3, we can see that with an 18% increase in chip area, the execution time decreases by 54% with the FSME. Similarly, for fast motion estimation the execution time goes from 20Million cycles down to 16Mcycles per macro-block. We now instantiate this framework with two target styles: FPGA and ASIC. More info in[145].

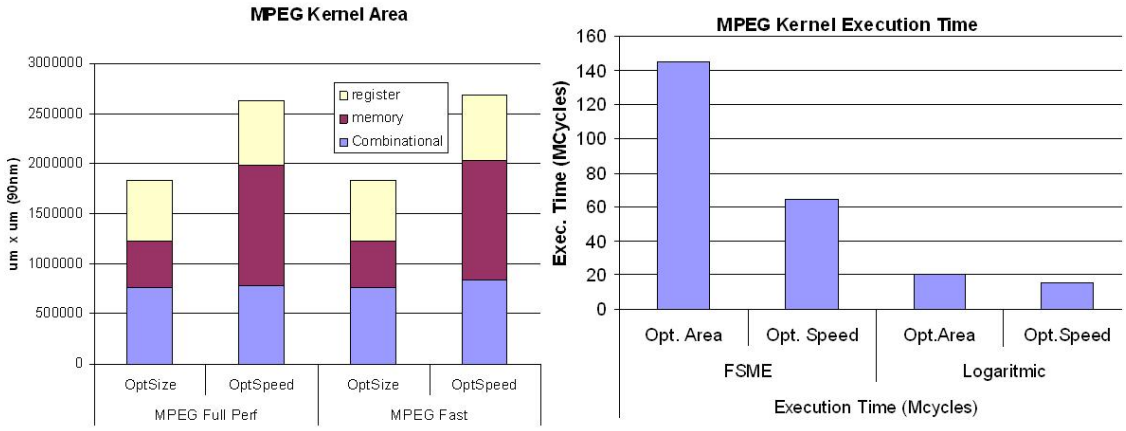


Figure 4.3: MPEG Kernel optimized for area versus optimized for speed

4.3 FPGA platform

We decided to target the MPEG kernel on the Altera-FPGA PCI Express Development Kit, Stratix II GX Edition [62] as a HW-SW platform for reconfigurable systems and ASIC prototyping. Altera's NIOSII [63] soft-core lets designers to integrate custom instructions, hence system designers can fine-tune the hardware micro-architecture to meet performance goals and also the designer can easily handle instructions as macros in C/C++. We modelled the NIOSII with capacities similar to the ones from ARM926 [62]. The synthesized NIOSII has a data/code cache of (8K/8K) similar to ARM, a hardware multiplier/divider. It has a 6-stage pipeline with branch prediction. We have not added a Floating Point Unit (FPU) because our computation does not need floating point calculations. As peripherals, we selected a SDRAM controller, a SRAM bridge, on chip RAM and a performance counter (only for debugging purposes). For our comparison, we decided to target the MPEG kernel on the QuartusII-FPGA PCI Express Development Kit, Stratix II GX Edition [10] as a HW-SW platform for reconfigurable systems and ASIC prototyping. FPGAs are slower than their application-specific integrated circuit (ASIC) counterparts and draw more power because the number of transistors per logic gate is higher and because interconnections use switches and this leads the chip to consume higher static and dynamic power. Some of their advantages are a shorter time to market, as they save the ASIC back-end processes, the ability to re-program in the field to fix bugs, and lower non-recurring engineering (NRE) costs. Sometimes, designs are developed on regular FPGAs and then migrated into a fixed interconnection version that resembles an ASIC, called a structured-ASIC.

4.3.1 Metrics and Energy Estimation Model

The RTL verilog code description that comes out from Cynthesizer is passed through a synthesis tool (Synplicity) with the Altera StratixII libraries. Then, we obtain another

Verilog description with technology information about the FPGA. This description is loaded in Quartus II to synthesize its platform. EDA tools provided by the FPGA vendor offer a tool suite for power analysis and optimization that allows estimating device power consumption and even heat dissipation from early design concept through design implementation [146]. Optimizing the register transfer level code generated from the behavioral synthesis tool (section 4.1) can be a tedious and difficult process since the automatically generated code is complex and the interpretation is tricky and dull. Hence, manual optimizations at this level are quite difficult since they have been already made at higher level (behavioral SystemC). The produced code is prepared automatically accordingly, and its reading is knotty. We have compiled and programmed the NIOSII and the C code in the platform.

Communication between kernel part and control part is done with FIFO's. Following a previous work [147]. In this work, high level FIFO channels (such as SystemC *sc_fifo* or tlm_fifo channels) are automatically replaced by real hardware FIFOs. This approach provides a mechanism to communicate those parts of the common description of the system that will be implemented in SW by a processor with the pieces that will be synthesized to hardware, without any modification of the system description structure and preserving blocking synchronization. In the proposed system there are two FIFOs with a macro block size customized to the MPEG characteristics; one connects the kernel part (Q-zigzag algorithm) and the Huffman algorithm that is computed on the NIOSII.

From the processor perspective, the FIFO is shown in its memory map as unique address position, or equivalently, as a device with only one register. So, block transfers are done simply by consecutive writes or reads to this position. The blocking mechanism is handled connecting the *wait_request* signals of the NIOSII system bus with full-empty signals of the real FIFOs. From the kernel point of view, just after synthesis stage, high level FIFO channels are replaced by a RTL wrapper designed for synthesized modules to access the FIFOs, and maintain the blocking mechanism. Huffman is computed in the processors and results are written in another memory space that is another FIFO with similar size, where the compressed image is produced and afterward, sent again to a L1 memory.

4.3.2 Platform Dependent Optimizations

Manual optimizations at this level were carried out to manage L1 memory (from image to macro block size). As a result, the amount of memory required to implement the MPEG compressor has been significantly reduced thanks to the DTSE transformations. After different optimizations, the memory needed by the application is not too high and the design fits in a medium size FPGA. This transformation was carried out at SystemC level, taking into account the size and structure of the different MPEG memories and according to the Altera RAM block types (M-RAM, M512, M4K and LCs). Available memory bandwidth has to be as high as possible in order to compute the maximum number of pixels in parallel that loop unrolling permits. Unrolling DCTs makes different multiplications run in parallel. We fitted these multiplications in the FPGA MAC blocks (y=a*b+y'), which accelerated their computation. Table 4.1 shows the main features of

MPEG implementation (Family Stratix II Device)	Sw impl. NIOSII FSME	FSME Opt.Size	FSME Opt. Speed	Fast Opt.Size	Fast Opt. Speed
Combinational Aluts (K)	3,8	48(4)	65 (4)	48 (4)	66 (4)
Dedicated logic Registers(K)	3,1	46(3)	60(3)	46 (3)	59(3)
Total Registers (K)	3,1	46 (3)	60(3)	46(3)	59(3)
Total blocks memory bits(K)	287	312 (287)	352 (287)	310(287)	364 (287)
DSPblocks-9bits elements	4	2 + 4	4+4	2+4	4+4
Total Thermal Power Dissipation (mW)	495,4	1287,72	1288,09	1288,01	1288,01
Execution time (M Cycles)	2433,4	88,4	38,58	18,1	14,5
Max work frequency (Mhz)	207,56	43,34(207,56)	41,82(207,6)	43,54(207,6)	43,0(207,56)
Energy IPBB (J)	5,80	2,6	1,2	0,5	0,4

Table 4.1: Size and power numbers of TOTAL implementation and control (in brackets) MPEG parts, execution time is given for a IPBB GOP

the MPEG kernel implemented in an Altera FPGA. First column shows that the results for the "completely software" solution , both the kernel and control part are loaded on a NIOSII. We observed that the execution time is too high . In the second column we observe the values for a MPEG implementation with the FSME algorithm optimized for size, i.e. without loop unrolling. Its size is smaller (registers and memory blocks) but the execution time is higher. Third column shows the implementation optimized for speed, with a larger size and shorter execution time. There are similar trade-offs in fourth and fifth columns (where fast ME is used), as well as in second and third.Four MAC blocks are needed for the FPGA implementation. If the implementation is optimized for size, only two extra MAC blocks are required. Power consumption for the completely SW implementation is 0.4W and for the rest is 1.2W approximately. Figure 4.1 shows an increase of 144% in the number of ALUTs, and a 72% increase in memory that leads to a 55% computation time reduction. The percentage of improved cycles is similar to the ASIC case. FPGA consumes similar amounts since the static energy is dominant over the dynamic energy.

NIOS II software optimization

In the NIOS II architecture it is possible to control the debug properties ((flag –O0) to most (-O3) and the possibility to optimize for memory code size (-Os)). It can be configured in mode debug or in mode release. There is also the debug level that can go from none to the maximum (-g3).In our experiments, we obtained the number of clock cycles (Millions) with the DMA code and the entropy code for a QCIF (176x144) image. From a non optimized version to a complete optimized one, we observed a decrease by 84.6% for DMA and 54% for the entropy encoder in the number required of cycles. This substantiates our claims that sufficient effort in source code transformations is essential for making reasonably fair comparisons between different platform mappings

and styles. In table 4.1, we observe two implementations of the kernel: one optimized for size (a.k.a Opt Size) and one for speed (a.k.a Opt Speed). The Opt Size implementation has 48K ALUTS and 46K registers whereas the Opt Speed has 35% more ALUTs, 30% more registers and 13% more blocks memory bits. The power consumption is very similar between the two implementations since the static power dissipation of a FPGA is predominant compared with dynamic power. The number of DSP blocks in the Opt Speed is higher than in Opt Size, since more multiplications are made in parallel. The maximum frequency between both implementations is very similar. The control part was implemented on a NIOSII with similar architecture to the ARM used in 5.1 and can run slightly above 200Mhz. The energy consumption due to the control part is the 38% of the whole implementation. The ALUTs used by the control part is less than 10% of all system but the control part has a dedicated memory larger than the kernel part; from 11 times for the solution Opt. for Size size to 3 times for Opt. for Speed. This is due to different reasons: first, the control part has not been refined with any DTSE optimizations; furthermore, this part has all the entropy encoding tables to produce the output stream; and finally, the NIOSII memory size would permit to add new features to the entropy encoder module. Execution cycles are calculated for a group of pictures of 4 frames (The image size is the standard QCIF equal to 176x144 pixels.) one I one P and 2 B (GOP IPBB).

4.4 ASIC Framework

Cadence's BuildGates [148] tool was used for ASIC synthesis. Using the technology logic cell library, the synthesis tool performs the process of transforming the ASIC register-transfer level (RTL) description into a technology-dependent netlist. The netlist is the standard cell representation of the ASIC design at the logical level. It consists in the standard-cell library element instances, and port-connectivity between them. Proper synthesis techniques ensure mathematical equivalency between the synthesized netlist and original RTL description.

Since both logical and netlist views are only useful for abstract (algebraic) or approximate-time simulation, but not device fabrication, the physical representation of the chip must be designed as well. This is also called the layout view, and is the lowest level of design abstraction in common design practice. From a manufacturing perspective, the standard cell's VLSI layout is the most important, as it is closest to an actual "manufacturing blueprint" of the standard cell.

4.4.1 Metrics and Power Model

For the FPGA implementation, a Verilog output description from Cynthesizer was loaded in the Synplicity tool with the libraries for StratixII FPGA. Here, the synthesis from the Verilog register transfer level previously generated with Cynthesizer is used by BuildGates to generate a Verilog structural description in which the basic elements are cells (standard cells and macrocells) belonging to the library designed for the target technology. The script also has the call to the file where the Verilog register transfer

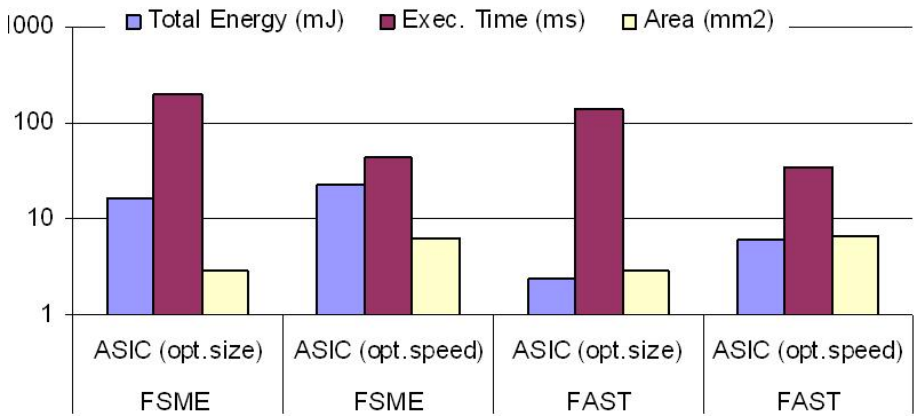


Figure 4.4: ASIC total energy, execution time and area of four different MPEG implementations

level MPEG code is. Executing a *do_build_generic* command the tool creates a MPEG structural description with all the elements from the UMC Faraday technology, after a *do_optimize* command with the flag *–power*. We carried out a logic simulation with the netlist with Verilog-XL. We also obtained a timing report (*report_timing command*) that makes the maximum gate delay available for the worst case scenario. This maximum clock frequency was obtained without parasitic elements of a post-layout simulation. The Verilog code obtained is then loaded in the First Encounter tool. First Encounter produces the ASIC layout by transforming a structural circuit description through several stages down to a geometrical description (design layout).

At this abstraction level, all modules are hardware descriptions and therefore a complete exploration of all nodes is possible. BuildGates provides a complete power report when the synthesis is completed. The tool provides information about the internal cell, nets, and leakage power. Memory power is obtained from the datasheet that provides Faraday when they are created. Then the energy per activation and leakage power for the MPEG were obtained with a gate level simulation with Verilog-XL.

4.4.2 Platform Dependent Optimizations

We carried out several DTSE transformations during the MPEG design time to fit the code structures into the L1 memories. Therefore, the percentage of on-chip memory integrated with respect to the global requirements of the MPEG, ranges from 25% for the implementation optimized for size to 45% for the implementation optimized for speed. Architectural exploration analysis at this level using simulations (for verification of results) is undoable due to the extended time necessary to carry out simulations. Simulations that take seconds at TLM level (as for example the compression of an image) would take days at RTL level. Therefore, the simulation must be carried out with a very small image size, for example using instead of the whole image a single macro block. The synthesis layout results can be seen in figure 4.5.

In our design, we built an ASIC for the MPEG kernel part in 90nm UMC Faraday technology (Faraday refers to the cell libraries used for the UMC technology). We fulfilled the placement and routing of the chip using Cadence tools. Further verification must be done around the back-end process, such as pre- and post-layout simulations, to ensure large enough fault coverage (around 95% of nodes verified with the test vectors).

Using the technology library's cell logical view, the synthesis tool performs the process of transforming the ASIC's register-transfer level (RTL) description into a technology-dependent netlist. The synthesis from Verilog register transfer level previously generated with Cynthesizer, is made using BuildGates to generate a Verilog structured description where the basic elements are cells (standard cells and macro-cells) belonging to the library designed for target technology. First Encounter produces the ASIC layout by transforming a structural circuit description, through several stages, down to a geometrical description (design layout). Results can be observed in figure 4.5. The percentage of on-chip memory integrated with respect to the global requirements of the MPEG ranges from 25% for the implementation optimized for size to 45% for the optimized for speed.

Figure 4.4 illustrate a bar-chart, providing information about total chip energy, execution time and area. There are four different solutions, first ASIC optimized for size (restricted number of FUs to compute each algorithm), ASIC optimized for speed with several functional units to compute in parallel the algorithms. The figure 4.5 shows the layout of the kernel part. The UMC faraday technology permits 9 metal layers. We can observe that the memories are distributed around the chip and the standard cells are located in the middle of the chip. The final die size is 1.7mm^2 for the implementation optimized size and 2.5 x 2.5mm^2 for the implementation optimized for speed. This ASIC can run up to 450MHz. The control part is based in the ARM926 EJ-STM, that is a fully synthesisable processor. It is available trough the processor foundry program. Other features are provided in reference [149]. The amount of computation in the kernel part is negligible in comparison with the kernel part. Therefore, the ARM at this level can be seen as a black box. The communication between the synthesized part and the ARM would be similar to the solution of the FPGA with the NIOSII. If we would like an ARM implementation. There are two ARM possibilities that are shown in reference [62].We have studied the ARM version optimized for size because the control of the applications does not need too tight requirements. The CMOS sensor works at 96 Mhz, so that ARM running at 250Mhz instead of the ARM at 450Mhz with higher area, has enough performance to manage incoming data with the help of the DMA while processing the VLC algorithm. Then, the power consumption due to the control is not as high as it would be with an ARM with high performance. Therefore, there are two different clock domains: the MPEG coprocessor (kernel part) and the ARM. The FIFO size to synchronize both domains is 8x8 32bits block length. The communication between the coprocessor (kernel part) and the ARM is very limited because the ARM has to compute 10% of the all compression. Data fetched between coprocessor and ARM is just the DMA data, control signals, and the DCT output data that has to be afterward VLC computed in the ARM. Hence, the ARM selected is the ARM optimized for size, with an area of 0.85mm^2. ASIC with an IPBB GOP qCIF size (FSME algorithm) simulation

(b)

(a)

Figure 4.5: MPEG ICs layout in same scale (90nm technology), (a) optimized for size (area 1,7x1,7mm^2@450MHz) and (b) optimized for speed (area 2,5x2,5mm^2 @450MHz)

consumes 27mJ for the ASIC opt. for size solution and 38mJ for the ASIC opt. for speed solution (FAST algorithm 4mJ and 10mJ respectively). The area is 2.89 and 6.25mm2 and takes 106Mcycles and 48Mcycles respectively.

4.4.3 Summary

In this chapter, we focus on customizable platforms: FPGA and ASIC. Each platform framework has been explained. And also the metrics and power models used. Furthermore, we give details about the platform dependent transformation attained to the compressor. Basically, two syntheses have been carried out. A Basic one without loop unrolling for a minimal area and other one optimized for speed with higher area. This has been made looking at each algorithm involved as well as the computation and necessary area to compress the different frames. We used parametric FIFOs with different layers of abstraction to connect the hardware (kernel) part based on a data-flow computation with the embedded processor that is the responsible of the control part. We also show MPEGs IC layout of both ASIC versions. In the next chapter we are going to make something similar but for platforms based on Instruction Set Processors since they are DSPs or ASIPs.

5 Instruction Set Processor platforms

A unique C++ model is directly translated to a C description to be mapped in the DSP and ASIP platform. Even if this C description has special characteristics (such as: no pointer or not dynamic memory allocation), the code is still not fully optimized and some optimizations are needed to extract the maximum performance of the application. Both implementations, DSP and ASIP, have an Instruction Level Parallel compiler that does standard compiler optimizations. Compilers targeting the embedded domain have strong restrictions on area, energy and performance and optimizing compilers has become an essential component of high-performance and embedded computer systems. An extensive bibliography on traditional compiler optimizations can be found in [150, 151, 152].

Besides the usage of compiler optimization, since multimedia applications are based on intensive data computation, the reduction of transfers between different memory levels can considerably reduce energy consumption. Data Transfer and Storage Exploration (DTSE) optimizations [153, 154] aim to improve spatial and temporal locality to minimize the load of shared buses which is the main source of power consumption [155]. These transformations are usually achieved at a high expense on addressing and local control. Some source code optimizations reduce control flow. Control flow optimizations are based on mathematical models combined with genetic algorithms and modify source code trying to minimize the number of iterations that control-flow statements evaluate. Loop nest splitting minimizes the number of executed if-statements in loop-nests of embedded multimedia applications. Advanced code hoisting moves portions of the inner loops to outer ones. Ring buffer replacement eliminates small arrays serving as buffers for temporary data. These three techniques reduce control flow, arithmetic calculations and execution time and hence, they minimize energy consumption ([156, 157, 158]).

5.1 Digital Signal Processor - Based Platform

There are significant differences between the different families of DSPs and a good election is highly decisive. Due to the character of our code, as a representative DSP-based platform, we selected the Texas Instruments DaVinci platform [159]. The DaVinci solution is a DSP-based system-on-chip (SoC) platform optimized to fulfill the performance and feature requirements for a broad spectrum of digital video applications. More specifically in our case we used the TMS320DM644 [121, 160] which, in addition to the DSP core, has an ARM926 [62] processor. The core of the DSP is a VLIW processor core specially designed to exploit instruction level parallelism and to maximize channel density in I/O chip communications. It contains two identical clusters with four FUs each and represents a typical clustered DSP.

5.1.1 Framework

The Code Composer Studio [161] is the development framework (IDE) provided by Texas Instruments for the TMS320DMxx family. The Setup CCStudio provides an interface with the development board that contains the DSP. Independently of the board used, the framework permits the processor, the working frequency and external memory on which the application will run to be configured. The Digital Video Evaluation Module (DVEVM) [162, 160, 161, 121, 163] enables developers to immediately evaluate the DaVinci processors and begin building digital video applications. The DVEVM allows developers writing production-ready application code for the DSP and ARM926EJ-S.

5.1.2 Metrics and Power Model

The code memory requirements, after an accurate DTSE exploration, ensure that the required memory structures are small enough to fit them all in the DSP L1 internal RAM. The kernel part is loaded in the DSP while the control part is loaded in the ARM (similarly to the custom implementations). The bandwidth of the coded memory structures is large enough to allow the DSP to use all its potential, that is, the eight Functional Units (FUs) with software pipelining.

The power consumption of TMS320DM6446 [163] can vary widely depending on the use of on-chip resources. Thus, power consumption cannot be accurately estimated without an understanding of the components of the DMSoC in use and the usage patterns for these components. By providing the usage parameters that describe what is being used in the DMSoC and how accurate consumption numbers can be obtained for the power-supply and thermal analysis. This model breaks down power consumption into two main components: static and dynamic. Using this model, different applications that use the DMSoC differently can obtain accurate predictions over the entire spectrum of possible power consumption in DM6441. To use the spreadsheet, simply select the case temperature for which you want to estimate power and fill in the appropriate module use parameters. The spreadsheet takes the provided information and displays the power consumption details for that configuration.

5.1.3 Platform Dependent Optimizations

Usually, programmers write code following the model or functionality and make the code as readable as possible. For a SW designer using a DSP, the easiest method to optimize the application is to let the compiler handle their different optimizations. In fact, these optimizations result in a huge reduction in the execution time, and hence consumption. Among other traditional optimizations ([150, 151, 152]), the compiler is in charge of analyzing the different loops and determining if unrolling certain loops is worthwhile, in which part of the code there are good candidates for software pipelining, etc. For each analysis, the compiler must evaluate the factor that the code will improve. Nevertheless, the results achieved with the compiler are not always the best possible results. This can occur for two reasons: first, good “human-readable” code is usually not so good for the compiler because it can hide possible optimizations; and second, the compiler does not

have enough information for carrying out more aggressive optimizations and uses more conservative solutions to assure that the algorithm is executed correctly.

SIMD instructions: Some Texas Instruments DSPs are capable of using Single Instruction Multiple Data (SIMD) operations. SIMD operations are instructions employed to achieve data level parallelism in a similar way as vector processors. SIMD operations in Texas Instruments compilers are based on intrinsics. Intrinsics are functions handled especially by the compiler and usually substitute a sequence of automatically-generated instructions, in which the compiler has an intimate knowledge of the intrinsic function and can, therefore, optimize it better. The inclusion of SIMD instructions can be automated by the compiler; however, in most cases hand tuning of the code is needed since the compiler does not have enough information to detect when it is possible to use SIMD.

In house libraries: TI, as most DSP vendors, supplies libraries specialized in several aspects, such as the IMGLIB that provides some functions commonly used when working with image manipulation. Among these, there are functions that implement IDCT, FDCT, quantization, etc. Using these functions leads to a considerably faster execution. Basically, this library (and related functions) exploits maximum SIMD instructions and uses the maximum bandwidth of the data buses. To use these functions correctly, the parameters for passing data have to fulfill certain requirements: data passed to the function must have a specific format and should be aligned in the memory. If data types of the algorithm are not compatible with the required format, these functions cannot be used.

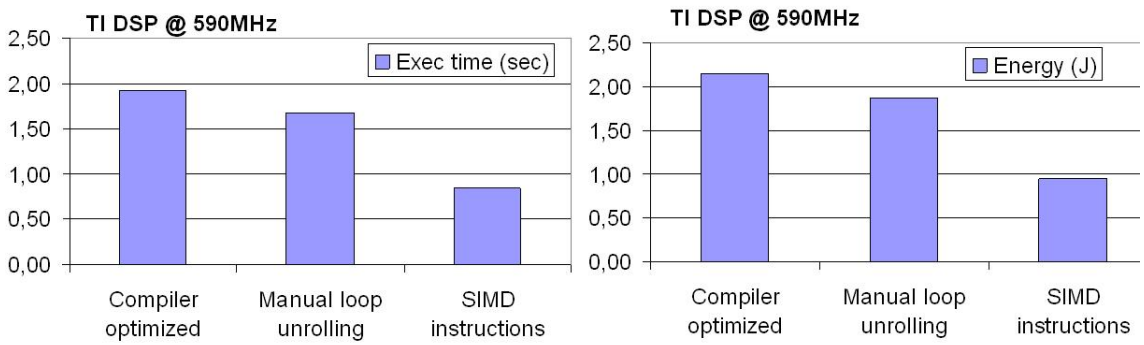


Figure 5.1: Execution time and energy consumption for the TI- DSP after different optimizations

Application code for the ARM926 has similar characteristics that ARM926EJ but it has 8K/16K instead of 8K/8K and works at 297 Mhz instead of 250Mhz. Other features are similar. As already explained the most computing intensive algorithm is the ME with the SAD computation. Therefore, we made a hand made loop unrolling to help the

compiler for software pipelining. This caused an improvement reducing a 16% the needed cycles. Before using instructions that takes profit of SIMD architecture, the bottleneck was sharing data between on-chip memory and FUs. The use of SIMD instruction set permits to make more than one operation simultaneously on data vectors. An example is DOTPU4*src1* , *src2* that realizes multiplication with 4 operands vectors between operands *src1* and *src2* and afterwards realizes the scalar addition of the results. With those instructions, the number of executed instructions is reduced while the number of read writes to memory is kept constant. With this, the bottleneck remains exclusively in the data memory and data cross paths, responsible of moving data from the register file to the functional units. This happens always with the ME computation where execution requirements between blocks are few operations but very intensive making that ME has 60% time of total coding. More intensive use of SIMD instruction can be made in the code. For example, using the DCT function that is part of the TI library. Among other traditional optimizations ([150, 151, 152]), compiler is in charge to analyze the different loops and to observe if it is worthy to unroll certain loops, what parts of the code are good candidates for software pipelining, etc Figure 5.1, shows the influence on energy of the different optimizations carried out. The comparison begins with the compiler-optimized code since any engineer can activate the maximum possible compiler optimizations. As seen in the figure, manual loop unrolling and the inclusion of SIMD instructions can halve the execution time, which implies a similar reduction in the energy consumption. Actually, in Figure 5.1 it is observed that in the version optimized by the compiler with manual unrolling slowing decreasing the execution time by 11%. After this, we can still improve the execution time by another 53% by applying SIMD instructions. In total, applying manual loop unrolling and the SIMD usage, we achieve a 64% improvement in execution time and a reduction of 60% in energy consumption.

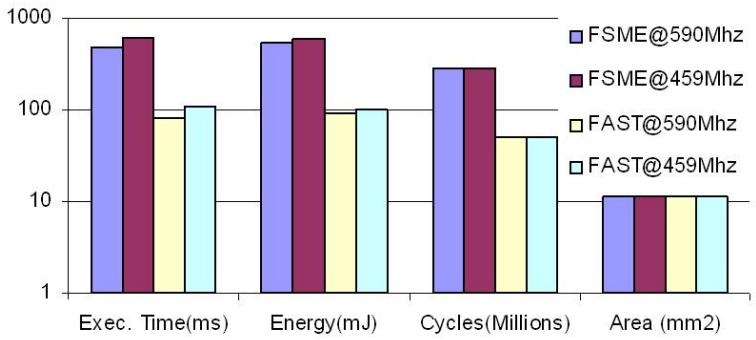


Figure 5.2: Total energy, execution time and area for the TI-DSP after different optimizations

The DaVinci implementation is completed using two DSP clock speeds (459Mhz and 590Mhz). The execution time for a QCIF size of 4 images GOP IPBB is 1,92 and 2,56 seconds respectively, the energy consumed is 2,1 and 2,3 J. All the information is extracted from the chip maker spreadsheet [163]. The DaVinci area information comes

from references [164, 165]. In figure 5.2, we observe the different executions times, and energy for DaVinci implementations. We observe that computation (energy, execution time) is much higher with FSME algorithm. Since the implementations are made in the same platform their area is the same.

5.2 Application Specific Instruction Set Processor- ASIP

An ASIP is a processor dedicated to one application or domain of applications. It has a customized instruction set or specific hardware resources and requires an ILP compiler. Since ASIPs are optimized towards certain applications, they combine high performance and efficiency of a dedicated solution with the flexibility of a programmable solution, and hence they fill the gap between highly optimized platforms like ASICs and the more flexible solution offered by DSPs. Developing an ASIP requires a large hardware and software design effort, but it can be reused easily, since new versions of the product only need to compile the new source code.

5.2.1 Framework

Application Specific Instruction Processors (ASIP) are domain-oriented processors in which the instruction set is tuned for a specific application. Some ASIPs have a configurable instruction set and these cores are usually divided into two parts: static logic, which defines a minimum ISA, and configurable logic, which can be used to design new instructions. Low energy is one of the key design goals of the current embedded systems for multimedia applications and Very Long Instruction Words (VLIW). ASIPs, in particular, are known to be very effective in achieving high performance with reasonably low power, which is the main objective in the domain of interest. For our experiments we used a retargetable VLIW-ASIP compiler and simulator framework based on Trimaran [58] called COFFEE [118]. This framework can map applications to a broad range of processors and memory configurations with different instruction set architectures and can also simulate and report detailed performance and energy estimates. Possible compiler research optimizations for these architectures are high-level machine independent code transformations and "back-end" machine dependent optimizations such as instruction scheduling, register allocation, etc. The architecture used for this work is VLIW-based and, as for all VLIW processors, the processor executes instructions that are composed of parallel operations that are executed in the different Functional Units (FUs) (sometimes using coarse-grain reconfigurable logic). The processor's data path is divided into clusters (or slices), which contain one or more register file, one or more functional unit, and bypass logic. This well known division of the data path into clusters reduces the energy consumption and the data path delay by reducing the complexity of the register files and the bypass logic. Since reducing the number of cycles is also an effective way of reducing the energy consumption, all data path operations can be predicated to enable efficient execution of loops with conditionals. Furthermore, chains of data dependent operations can be executed in a single clock cycle, using software controlled functional unit chaining to decrease the effective latency of a set of operations, and hence reduce

the number of clock cycles. Thanks to this chaining, some register file accesses can be prevented, which reduces the energy consumption further. The COFFEE framework allows broad compiler research and instruction set exploration to obtain an optimal ASIP architecture for the required application.

5.2.2 Design Metrics and Energy Estimation Model

At this abstraction level, the complete hardware description is not yet needed and therefore exploration can be faster. To obtain early estimates with sufficient accuracy, we carried out the following methodology: Different instances of the components of the processor (Register File, ALU, pipeline registers, etc.) were designed at RTL level with an optimized VHDL description. In our case, for each instance, logic syntheses were carried out with the UMC@90nm general purpose standard cell library from Faraday [144], also used for the ASIC power model. The result of the process is a library of parametrized energy models. Energy per activation and power leakage for the different components are estimated from the activity information from gate level simulation and the parasitic information. Memories used for this analysis are highly optimized custom hardware blocks and hence the standard cell flow cannot be used. We created a library of memory blocks with the corresponding energy consumptions (dynamic and leakage) using a commercial memory compiler (from Artisan). Finally, our pre-computed library contains the energy for various components of the processor using standard cell flow, and for memories, using the commercial memory compiler. A detailed description of the complete energy model is described fully in [118]. Figure 5.3 shows more impressive values than in the other platforms since in this case the HW and the SW are dedicated completely to the application. Then, in terms of energy, the DTSE transformations are improved by 25%, control flow and dedicated instructions are improved by another 58% and the AGU optimization by another 14%. In total, we obtained a 97% improvement. In cycles there was a similar improvement of 95%.

5.2.3 Platform Dependent Optimizations

In addition to the optimizations explained in Section 5.1.3, we carried out a generic architectural research exploration. Certain optimizations were studied for the ASIP platform. First, multiply-and-accumulate operations and divide operations, very popular in multimedia applications, can be accelerated using specific hardware support. The needed hardware and compiler support will improve energy reduction significantly, and also boost the speed up to decrease the number of cycles needed for the application. Once the memory and data computation have been optimized, the bottleneck resides in generating the application code addresses [166]. Address computation is extremely important in multimedia systems and often involves linear and polynomial arithmetic expressions that have to be calculated during program execution under strict timing constraints. Since multimedia applications are based on intensive deep-nested loops in which the majority of the code is executed, a specialized address generator unit was developed to speed up all address related computations in the inner-most loops. Address

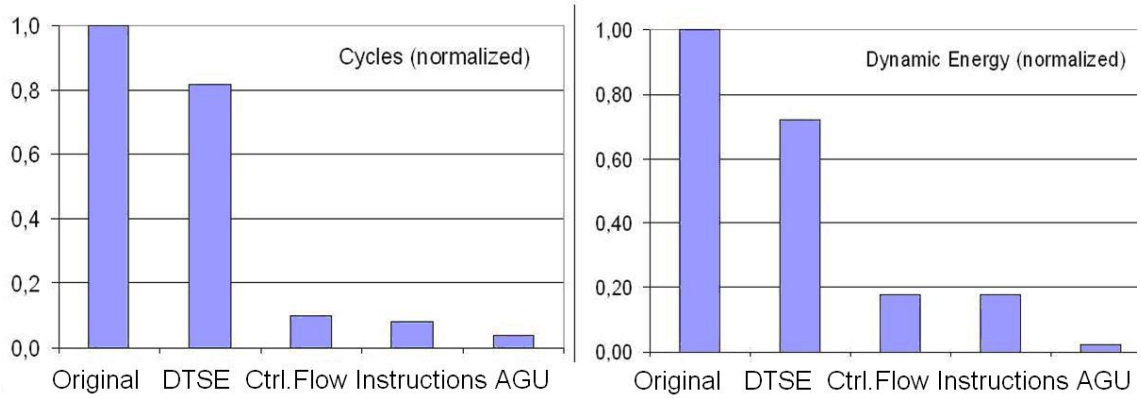
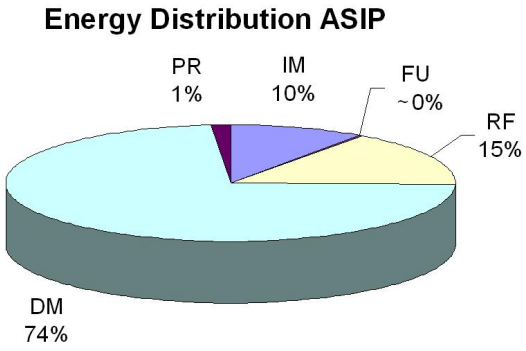


Figure 5.3: Energy a) and speed up b) improvement after different optimizations for a fixed processor after general data-path and memory exploration.

generation in these parts of the code is easy and does not require a lot of hardware support, hence minimum hardware and a small number of registers are sufficient to generate the addresses of the inner-most loops. Addresses for the rest of the application are computed in the default (more complete) address generation unit, which is similar to a normal functional unit.

As mentioned before, Data Transfer and Storage Exploration (DTSE) [154, 153] transformations aim at the optimization of the accesses to the different memory layers of the hierarchy. Several steps of DTSE insert complex control flow and addressing code. Removing this control flow is crucial to get an optimal implementation of any program. Different techniques have been introduced and are broadly explained in reference [117]. Scratch pad memories (SPM) are high-speed memories where the compiler generates explicit instructions to move data from and to the following levels of the memory. SPM have been proven to be more energy, area and performance efficient than caches [167, 168] and they are an optimal choice for embedded systems. Moreover, in any typical multimedia application, significant amount of execution time is spent in small program segments. Energy can be reduced by storing them in a small loop buffer instead of the big instruction cache *[169, 170, 171, 172]*. The data path can be composed of one or more clusters. It contains one or more register files and one or more FUs. By clustering, the routing length and interconnect complexity inside a cluster are reduced (i.e. good for both power and speed), at the price of increased compilation complexity due to the additional cluster-to-cluster data transfers. To get the optimal configuration of the data path a research exploration was done [173]. The application under study has some recurrent operations that consume excessive cycles . We added hardware support for multiply-and-accumulate and divide operations. Once all this costly instructions have been optimized, the bottleneck resides on the address generation [174] so a special address generation unit has been created to compute inner-most loop addresses.

*Global improvement:*After a general architecture exploration on the number of FUs,



	Dynamic	Energy (mJ)	%
DM	Data Memory	15,63	0,73
IM	Instruction Memory	2,07	0,10
RF	Register File	3,23	0,15
PR	Pipeline Registers	0,3	0,01
FU	Functional Units	0,07	0,00
Total		**21,3**	**100**

Figure 5.4: ASIP Total energy distribution in ASIP implementation for and average frame of an IPBB GOP

size of memories, loop buffer and register file sizes is done, these other optimizations improve considerably the results, boosting the performance and reducing energy consumption. Figure 5.4 shows the final energy distribution of an average frame for an IPBB GOP for the ASIP implementation. The analysis and comparison of these results with the other platform is realized in the following section.

5.3 Customized Platform vs ASIP Platform

The main aim of this thesis was to learn the lessons by accurately comparing different architecture mappings, explored during the implementation of relatively complex multimedia IP components. Our main test-vehicle is the MPEG-4 MP video encoder in 2 different forms: a rather regular one, and a more performance optimized hierarchical (but irregular) one. This encoder has been implemented in four different major platform styles. We discussed the platform-independent optimizations as well as the platform-dependent ones, which allowed us to compare the styles in an apple-to-apple way, and therefore provide the accurately calibrated relative impact of the different platform optimizations. In the customized class (chapter 4), we explained the HW-SW trade offs for the FPGA and ASIC platform styles. Customized platform solutions, such as an ASIC, permit complete hardware exploration and a dedicated embedded hard IP processor to be used for the software. The execution time can be decreased by 55% if a BASIC implementation (without unrolling loops) is replaced with an implementation better adapted to high speed computation, at the expense of increasing the chip area by 54% and the energy by 25%. There is also a trade off in the MPEG computation due to the ME algorithm involved. The FPGA allows to carry out a similar hardware exploration. In

this case the hardware must also fit in the cells and blocks of the FPGA, and the multiplications must fit the MAC blocks. L1 memory must be small enough to fit different FPGA memories (M-RAM, M512, M4K, and LCs). The control code is loaded into a soft core processor, NIOSII, which was used as the test-vehicle here. NIOSII is built with similar features to an ARM9. The max frequency in the kernel part is much slower (40Mhz) than its counterpart in the ASIC (450Mhz), while the software part in NiosII runs over 200Mhz. We also explored platform-dependent optimizations in the FPGA style, and obtained an overall execution time decrease of 55%, at the price of an increase in ALUT of 144%, and 72% additional memory bits. The energy consumption in the FPGA is really high (0.6J for MPEG FSME opt. size to 0.12J MPEG FAST opt. speed) because all transistors consume power and the static power consumption is predominant versus the dynamic power. The instruction set processor platform solutions, such as the TI DaVinci, have a DSP with 4 clusters with 2 FU each and an embedded ARM9 core. Therefore, the hardware is "rigid" and the embedded software has to be tuned to fit in the platform. The DaVinci platform is intended for high performance but has a large range of applications that are suited for more than just video compression. Therefore, the software has to be optimized and then loaded in the hardware with different optimizations: compiler optimizations, loop unrolling, and SIMD instructions usage. This leads to a considerable improvement in performance, decreasing additional, with respect to a naive implementation, by 60% in energy and by 64% in the execution time. This has been achieved by using all optimization resources offered by TI. In contrast, the ASIP platform solution is the most complex solution in terms of design-time, since both hardware and software must be tuned to fit the target application domain. However, these optimizations make the increase in performance much more impressive due to careful matching between software and hardware. We obtained a spectacular 97% improvement in energy consumption and 95% improvement in execution time. Note that the resulting ASIP is still instruction-set based and hence programmable within the specific target domain of video coders.

5.4 Target platforms Style Exploration: Set Up and Results

In our work, we have chosen four platforms widely used in the embedded domain. Any of those four platforms can potentially fulfill the requirements of our code with the required optimizations, but have advantages and drawbacks. In this section, we will first analyze the parameters used to evaluate the different platforms and after that, we will study each one of those platforms.

Nowadays, the compressor has a PSNR of 27 to 35dB for the foreman image series. A enhanced PSNR is achieved for I images and is degraded for P and B images. If we bear in mind that a PSNR up to 24dB is the one that a human can discern whether an image is visible or not We can conclude that these values are acceptable results. A bit far to be used in HDTV (higher than 45dB), but it is not a serious problem since it is possible to change structural computing parameters, like pixel bus width from 8 bits to 10 or 12bits, increasing MAC operations length and changing the tables to get more

precise values. This would produce more accurate results.

5.4.1 Model Trade-off design metrics

In our work we focus in the technological metrics of the different platforms under study. Those metrics can be evaluated in terms of costs and constraints. We have two different types of costs: implementation costs and design cost.

In the first category, implementation costs, we have area, speed and energy consumption. For a given technology, the area of a chip determines the cost of a VLSI process and this was historically the most important cost of a design. Nowadays, with the increasing capacity of integration the design area is not that crucial, but still has to be taken into account. In multimedia applications, the main contributions to this cost are due to the processing elements and memories. Today designs, especially in the embedded domain due to battery life, energy consumption is the limiting factor that drives design decisions and it depends on several parameters from physical to system level: technology, area, frequency, supply voltage, bit activity, algorithm, leakage, etc.

Design costs are related to the system complexity development: programmable solutions have the advantage of software developments over the slow hardware development of other solutions. An easy compilation and a retargetable framework is an advantage that can help to accelerate time-to-market of products or can facilitate upgrades or new versions of the applications. Constraints in the embedded domain are of two kinds: flexibility and performance. The flexibility of a chip determines the capacity to deal with different applications and the performance is a real time constraint related with the time needed to compute a set of operations. Besides this definition, in data-flow applications latency and throughput are also used as a measure related with time. The latency, in multimedia systems, can be expressed as the delay between data fetched from a sensor or memory and the processing of these data for its further exploitation (transmission, storage, display etc). Throughput is the amount of data successfully processed per unit of time, and is controlled by the available bandwidth, as well as the available signal-to-noise ratio and hardware limitations. In our case, performance is related with the time needed to encode a macro block and is linked with the latency and number of available resources and the communication between memories (where images are located) and functional units. Latency in our system is lower than in other Multimedia systems because we do not have to wait to have all images to process them since data is processed at macro-block level and not at image level with a similar throughput.

5.5 Platform Results Summary

We did a fair comparison mapping the same realistic application on different platforms with the same technology (90nm). Different implementations came up from the same original source code where we performed platform independent and platform dependent optimizations to exploit the maximum benefit from each targeted device, therefore, comparing results between optimized application-platform, making a unprejudiced comparison. Dependent on the available design time, area, performance, energy consumption,

		ASIC Opt size	ASIC Opt.speed	ASIP (standard Cells)	ASIP Macro	DSP (Da Vinci)	FPGA (opt size)	FPGA (opt speed)
Energy (mJ)	Memories	15,9	22,6	23,4	20,1	not available	not available	not available
	Leakage	0,5	0,3	0,5	0,2	not available	not available	not available
	"core"	0,04	0,04	2,6[1]	0,95[1]	not available	not available	not available
	Total	**16,5**	**22,9**	**26,4**	**21,3**	**536**	**2626,5**	**1188,3**
Time related	**cycles (in millions)**[2]	88,4/60,9 (14,4)	38,6/25,3 (14,4)	84,6/55,5 (14,0)	84,66/55,5 (14,0)	282,36/185	88,4/60,9	38,6/25,3
	Execution time (s)[2]	0,20/0,14 (@450MHz)	0,14/0,06 (@450MHz)	0,19/0,12 (0,03) (@450MHz)	0,17/0,11 (0,03) (@500MHz)	0,48/0,31 (@590MHz)	2,0/1,4 (@43,34MHz)	0,92/0,61 (@41,82MHz)
	Frame Rate[4] **(qCIF)**	15,3/22,2	35,0/53,3	15,9/24	17,7/27	6,3/9,6	1,5/2,2	3,3/4,9
Area (mm^2)		2,89	6,25	6,05[3]	2,42	11,4[5]	↑↑	↑↑
Design time	**First Implementation**	**HIGH** Synthesis front-end + back-end		**HIGH** (first version) ≈ ASIC development + compiler development		**LOW** Code compilation	**MEDIUM** Synthesis front-end	
	Following implementations	**HIGH** Synthesis front-end + back-end		**LOW** (following versions) Code compilation		**LOW** Code compilation	**LOW** Synthesis front-end	

[1]:the ASIP core contains the energy of the FUs and the register file.
[2]:the first number represent the cycles or execution time related to the kernel of the application (the number between brackets refers to the control part).
[3]:area for the ASIP based on std cells is an estimation.
[4]:first value GOP IPB second value IPP (IPB/IPP)
[5]:information obtained from the references [164] and [165] .

Table 5.1: FSME MPEG Summarized results

		ASIC (opt size)	ASIC (opt speed)	ASIP Std.Cells	ASIP Macro	DSP (Da Vinci)	FPGA (opt size)	FPGA (opt speed)
Energy (mJ)	Memories	2,3	5,9	9,7	8,9	not available	not available	not available
	Leakage	0,06	0,07	0,19	0,09	not available	not available	not available
	"core"	0,01	0,01	1,78[1]	0,87[1]	not available	not available	not available
	Total	**2,39**	**5,97**	**11,69**	**9,97**	**93**	**535,4**	**434,5**
Time related	**cycles (in millions)**	18,1/14,9 (14,3)	14,5/11,3 (14,3)	39,6/29,7 (14,1)	39,6/29,7 (14,1)	49/36,8	18,1/14,9 (14,3)	14,5/11,3 (14,3)
	Execution time (ms)	43/36 (@450MHz)	35/27 (@450MHz)	88/66(31) (@450Mhz)	79/59(28) (@500MHz)	83/62 (@590MHz)	416/342 (@43,54MHz)	337/263 (@42,98MHz)
	Frame rate [4] (qCIF)	74,7/90,3	93/119	34,1/45	37,8/50	36,1/48,2	7,2/9,6	8,9/11,9
Area (mm^2)		2,95	6,67	5,1[3]	1,73	11,4[5]	↑↑	↑↑
Design time	**First Implementation**	**HIGH** Synthesis front-end + back-end		**HIGH** (first version) ≈ ASIC development + compiler development		**LOW** Code compilation	**MEDIUM** Synthesis front-end	
	Following implementations	**HIGH** Synthesis front-end + back-end		**LOW** (following versions) Code compilation		**LOW** Code compilation	**LOW** Synthesis front-end	

Table 5.2: FAST MPEG Summarized results

flexibility and architectural requirements, designers can choose the more suitable platform style for their final implementation. The results shown in the following two tables 5.1 and 5.2 are for two different MPEG implementations: First implementation refers to the FSME algorithm and the second one to faster ME, with a logarithmic search. The former is representative of quite regular algorithms with a high access locality. The latter is representative of algorithms with a more irregular control flow and less local access, leading also to a more difficult address scheme. In both cases, for the FPGA and the ASIC, we did an analysis for a basic version, optimized for size, and another version optimized for speed. These two hardware implementations have been targeted to an FPGA and to an ASIC. The ASIP also has two different implementations: as seen in section 5.2. The framework used is mostly based on optimized macro-blocks which can be parametrized (a.k.a ASIP-Macro). This is the most likely style to use because an ASIP template (with parameters) can be heavily reused for different chip instances. So, the effort added to develop the macro library is well worth for the most critical blocks in the architecture like the data and instruction memories, register-files, and multipliers. The glue logic in between those macros will of course still be implemented with a conventional standard cell flow.

For comparison, we also give the results of the same ASIP based on a standard cell implementation (a.k.a ASIP-Std-Cells). Even if we recommend implementing the ASIP based on optimized macro-blocks, the ASIP based on standard cells gives a more direct comparison with the ASIC since we used the same technology running at the same frequency. In the following tables, we show the results in terms of energy consumption, execution time and area for the different implementations (ASIC, ASIP, DSP and FPGA). First of all, we should remember that, as explained through this chapter, the ASIC was implemented from a high level SystemC description of the application. Attending to our experience, the behavioral synthesis tool, even if it allows a very fast development, leads to a final design larger and slower than what an average engineer could achieve by direct manual coding in a hardware description language such as VHDL or Verilog (obviously after investing significant design effort). This implies that the design coming from SystemC (the ASICs and FPGA implementation) is larger, slower and has higher energy and leakage consumption than what a "good-hand-coded design" can achieve. Despite these results can be improved, they still give a fast and representative idea of a final project result where design time is limited, i.e. for most practical chip projects today. A remarkable result is already that the domain-specific programmable processor (the ASIP) and the standard-cell based ASIC platforms give, for each algorithm, results of the same order of magnitude, with differences depending on configurations (ASIP-macro, ASIP-std-cells, ASIC-basic or ASIC-optimized). As the application is data-flow dominated, the energy needed to access the memories is similar in the ASIPs and the basic implementation of the ASIC since memories are of the same size and technology and also the global architecture can be kept largely the same due to the ASIP style. That is one main difference with the more general focus of the DSP style. The optimized ASIC consumes much more energy in the memories since more memories are used. This boosts the execution time at the penalty of increasing considerably the energy consumption and chip area. The FSME algorithm is more computing intensive and the access to the

memories gives the major contribution to the final energy consumption for the ASIC and the ASIP versions. For this algorithm, the difference in cycles between the ASIC-basic implementations and the ASIP is of the same order of magnitude. For the Logarithmic ME algorithm the difference between the results of the ASIPs and the ASICs is higher. This is because this algorithm has much less data transfers and the contribution of the control has larger impact on the cycles needed to complete the encoding: ASIPs are better focused on data access and computing intensive applications and mostly suffer from a more irregular control flow. Some techniques such as predication alleviate the performance problem but increase the energy consumption. Other, more recent solutions have been proposed [175] which alleviate both the speed and energy problems but that are not yet available in the main stream commercial solutions. As we can see in [176], the usage of a distributed loop controller in addition with condition support improves considerably the performance and energy consumption (around a 30%).

5.5.1 Energy

The energy is lower in FAST than in FSME (table 5.1 and table 5.2) for all platforms because the computation needed to process the pixel data structures is inferior and control operations are similar. The main energetic contribution is due to the memories. In the case of the ASIC we take into account that all memories are switching at the same time. Therefore, this scenario is considerably worse than in ASIPs implementations where just the simulated switching was taking into account. The leakage of energy for ASIP and ASICs are similar: ASIC opt. for speed has more energy consumption because it trades-off power and execution time. The energy consumption is higher when the execution time is lower producing that the dynamic energy would be similar. ASIP macro architecture is more optimized and therefore the leakage power is lower. This is mainly due to two reasons: ASIP is running at a higher frequency (500MHz instead of 450MHz), and, finishing faster the computation, the time the design is leaking is considerably reduced leading to a half of the energy consumed (for leakage only) compared to the ASIC version. As mentioned in the previous paragraph, a hand-coded ASIC must give better results, in terms of energy consumption and leakage, so the difference with the ASIP version will be also reduced. To reduce the dynamic energy of the ASIC implementation we could decrease the voltage supply. This will lead to a reduction in the dynamic energy at the cost of decreasing the frequency and hence increasing the time needed to compute the algorithm. Also, we have to take into account that by raising time with the device working, the leakage increases. Therefore, it is not that worth to try to improve dynamic energy in the ASIC, at least not for the type of data-intensive applications considered here. The biggest difference between ASIP and ASIC implementations is in the energy of the core: the ASIP also contains the energy due to the register file contribution and this implies that this energy is several times higher than the ASIC which does not have a “power-hungry” register file.

The FPGA is, with a huge difference, the worst in terms of energy and clock speed. This difference is much higher than what has been reported in earlier comparisons based on simple kernels [128]. It is clear that even much optimized fine-grain cell alternatives,

as proposed also earlier in literature [129], would not be able to bridge this huge gap. There is the possibility in the FPGA to improve the energy consumption producing a structured ASIC with the HardCopy utility [177]. The ASIC produced can decrease in energy consumption around a 70%. Although, the produced structured-ASIC has decreased the energy consumption, it still has really higher consumption compared with the other implementations. The results on the DSP show that the core consumption consumes around a factor 10 or more than the ASIP implementation. This is mainly due to two reasons. First of all the platform dependent optimizations, basically the inclusion of specialized hardware (for the multiply and accumulate and division operations and the specialized address generator unit). Second, we have to emphasize that DSP memories are fixed and built for general purposes and for the ASIP case, we tuned all the memory hierarchy for this application (sizes, loop buffer, number of ports, etc.).

5.5.2 Time Related

The execution cycles in the FSME for an ASIC optimized for size is 88,4M very similar to the ASIP version (84,6M). A better result is given by the ASIC optimized for speed which gives 38,6M that is two times lower than the best ASIP approximation. The DSP results can be improved since we have used partially the low level assembler TI libraries to implement function of the video coder. The execution time is given in a worst case where the GOP includes an I frame, another P and finally a B frame qCIF(176x144) size 8 bits pixels. The values given in the table (in million cycles) are two: one for the GOP IPB and another for the GOP without B frames just a IPP, this GOP has more than 30% less computation and therefore has a considerable million-cycles reduction. Then, the frame rate has two possible numbers, one for the GOP IPB and another for the IPP. Between brackets (in million cycles too), there is the number needed to compute the control part; this computation can be done in parallel with the kernel part. We have to take into account that the kernel part is faster than the control. We have to improve, decreasing the needed cycles to compose the control computation, because we do not want them to be dominant in our application. When the frame rate is above 24 images per second we identify that we are working in real time. For the FSME, we get real time just for the ASIC optimized for speed solution. But for the FAST implementation we get real time implementation for all the solutions except the FPGA due that the working frequency is not too high. If we wanted to improve the frame rate it would be possible by producing GOPs with just I frames. The number of frames that we can achieve with ASIC opt. size and ASIC opt. speed for an III GOP (without prediction and bidirectional frames) is 123 and 203 images/sec (qCIF size). The corresponding number is 116 images/sec for the ASIPs macro solution, 113 images/sec for DSP and finally 27 images/sec for FPGA opt. for speed. Therefore, we can get real time for mobile CIF (352x288 pixels) in ASIC, ASIP and DSP. Remember, this coder computes in worst case scenario since all the macro-blocks are computed. If we changed the implementation, for example: when data or residual values are inferior to a threshold, then we can decide not to compute them. Then, we could increase a lot the speed of the solution but losing some quality performance.

5.5.3 Area

In ASIC area is doubled due to FUs parallelization that makes the speed to increase more than 3 times for FSME and 25% for the FAST implementation. This is due that in MPEG, ME is the dominant algorithm. By improving this algorithm, the whole system will be improved because it is, with a big difference, the part that needs more cycles to be computed. In contrast, with FAST implementation, ME is not as dominant as the rest of algorithms and the increase on speed improving this part is not as impressive as with FSME. The FSME ASIP solution has an area similar to the size optimized ASIC. The ASIP std. cells solution, the area is 5,1 to 6,05 mm^2 and the speed is half the one corresponding to the ASIC solution. The DSP solution has a higher area than ASIC and ASIP because the DSP is more "general purpose" than just a video encoder and has more resources not used for our work. The area in the FPGA is the highest of all the solutions because of its regular and repetitive structure and because of the configuration capabilities.

5.5.4 Design Time

The design time for the ASIC implementations is quite longer since hardware development is slow and the verification process consumes 70% of all the process. It is clear that this solution gives the best results in terms of energy consumption and in execution time (with the same clock), but the main drawback of this solution is that new versions of the design, to ad new functionalities or change the algorithms, needs to completely redo the design, with the cost on human resources, development time, etc. With more advanced deep-sub-micron technology nodes, the processing costs will become even higher, so the ASIC option is becoming very difficult to motivate, except for a few extremely high volume markets. Moreover, end-users must acquire a new device to benefit from the upgrades on the functionality. The development of an ASIP is even higher for the first design since, in addition to the design and development of the hardware, the ASIP requires a compilation framework. The main advantage is that this one-time effort is worthy since upgrades on the algorithms or new functionality just require to recompile the extended code. This is a witty advantage for the embedded domain, where totally new applications are limited, but they change quite fast with upgrades, new protocols, and more functionality. End-users can benefit from the upgrades, or even new features, just downloading a new version of the firmware. Mapping applications onto commercial platforms (DSP or ASIP) benefits of a really fast development since no hardware development is needed but the results are far from the customized platforms. Nevertheless, optimizing the source code improves this situation.

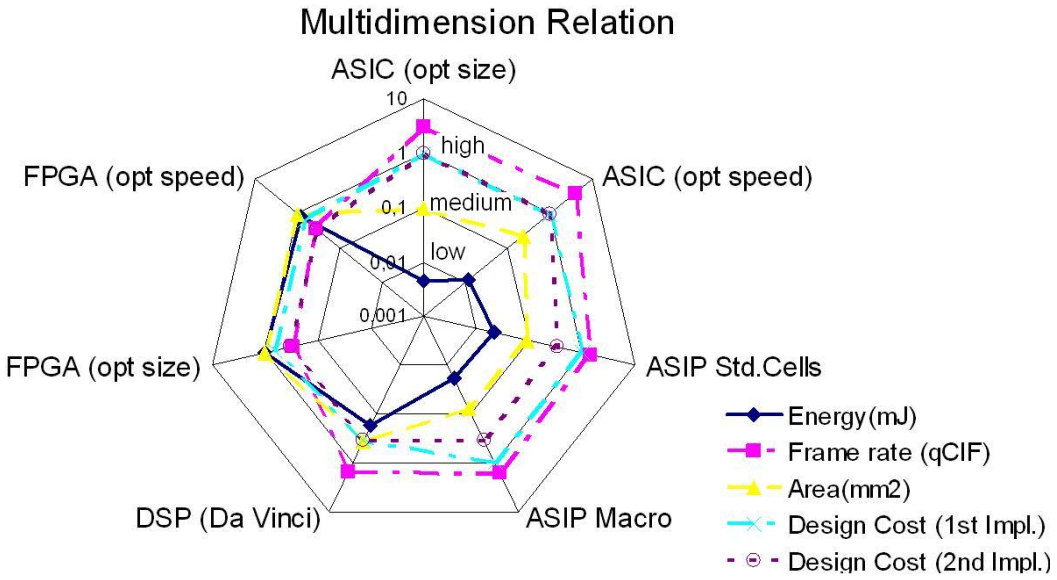


Figure 5.5: Multidimensional relation between implementations for the FAST solution

Figure 5.5 shows the comparison between the different FAST implementations for the higher value (energy, frame rate, area, etc.) and the rest of the solutions. All values are relative to the highest one except the frame rate where the maximum value is set to 24 images/sec. Consequently, frame rate values larger than one correspond to real-time solutions. If we move clockwise we are comparing a feature between different platforms, and if we move from the center to the border through the circle radio, we can see the different parameters of a platform.

The more generic DSP processors, like DaVinci, provide a quick implementation if the requirements in terms of the application (real time, energy consumption) are not very tough. If the application is more complex the FPGA solution must be considered again since this solution permits HW/SW full parallel and concurrent solution. But it must be stressed that the power consumption is the highest one, and therefore, it is not a good embedded solution when the battery life or power consumption form a limitation. As we expected, if you have time and resources, ASIC or ASIP solution can provide a more optimized solution.When the development time cannot be too high, an off-the-shelf solution, such as the DaVinci is the best one. FPGA solution is just for prototyping when some parts have to be accelerated with hardware and just for prototyping since power consumption is unacceptable for battery powered embedded systems . Some results can be obvious but numbers are extracted from the same source code and provide us a realistic view among implementations. This work is realized with a video encoder but this methodology can be extended to other multimedia space solutions such as 3D video games, or other fields that are not multimedia such as biomedicine, robotics or artificial intelligence where the technology requirements could be rather dissimilar. Other improvement would be to make this work not just for one task but for several

tasks since in this case the concurrency management would have given different results. An extended manuscript and software source encoder code can be downloaded from site [149].

5.6 Platforms Comparisons and Summary

One major objective of our work is to quantitatively identify all the elements needed to provide the system designer a very early estimation, and what platform style/subclass is the best choice to implement the target system. The results are exposed for a video compressor solution but they can be extended to any data-flow dominated system. Most multimedia applications nowadays fall into that class. In addition in this section, we provide actual comparative values among the different platforms and the results come from a single original description with sufficient effort spent on each of them to come to a sufficiently optimized solution. Therefore, these options are compared also in a fair way.

The image compression can be produced in very different situations with very different requirements. Therefore, all the solutions presented can make sense depending on the scenario. The principal purpose of this manuscript is discerning what solutions are the best in each case providing relative values among different possibilities.

Let's study the following three examples. In the first example we have to make the compression on a Wireless Video Capsule Endoscopy (WVCE)[178] camera that travels along the intestine to obtain information for medical inspection. In this case the image size is 256x256, the frame rate is two images per second and the battery life is 6 to 8 hours. The solution should be with very low power consumption and with very small computation requirements and compression should be without quality loss. The second example is a camera for video surveillance or domestic video. In this case, the video requirement would be 720x480 image size and 30 frames per second. At last, the third example is a future camera that will provide HDTV[179] (1920x1080 pixels) at 60fps.

In the first case, to perform the compression in a WVCE "Wireless Video Capsule endoscopy" an ASIC is used as medical device and it would be enough to compress the images. For medical requirements, compression can just be images I type without quantization (lossless compression) and this kind of process fulfills all our requirements of performance, power consumption, size and time to market. The ASIC solution is the one that provides less energy consumption, in addition, since we only need I frame images, a lot of chip energy due to memories is not used, therefore, the memories energy consumption go down to 0,1mJ (compare with table 5.2).

The battery has a storage capacity of 200W·h/kg [180]. The device has 2 batteries with 1 gr weight, consequently the batteries have a capacity of 0,4W·h. On the other hand the image sensor consumes 3 mW[181], the wireless transmitter 0,5mW·meter (-3dBm)[182], the 4 light Leds 1mW[183] each (they only emit light when a photo is taken).

The solution range must be till 100m, since the power consumption is the basic key and the wireless transmitter is the part that consumes higher with difference. With

these values we obtain that the power to realize these 2 images is 4mW in the ASIC opt. Size and 7mW for opt. speed solution. Then, the batteries last approximately 7 and 6 hours respectively. The ASIP solution makes the batteries last 4 to 3 hours, a period of time too short for ours purposes. Therefore, any other solution will be bigger and too power hungry (FPGA, DSPs) or really time consuming (and much expensive) to implement (ASIP).

In the second example, an off-the-shelf DSP can compress up to 30 images per second at a resolution of 720x480 with I and P frames and hence this is the most suitable solution since it has a fine compromise between energy consumption, area and design time and can handle the real-time requirements imposed. To achieve the same performance, an FPGA will be excessively power hungry and an ASIP or ASIC will demand too much development time, even if the performance and consumption were improved. The main problem of choosing a DSP is that this solution cannot be easily scaled if the requirements of performance augment. In this case, when a major change of the performance will be needed (for following versions of the product, for example) the project needs to be done again from scratch.

In this example, the most important requirement is not the power consumption but to know the maximum image size that we can compress in real time. Time to market can be important and for that reason we will choose an already developed platform, such as the DSP, where we can compress I and P frames. Image size range from GOP:III 663x542pixels@30fps to GOP:IPP 353x231pixels@30fps approximately. In this second example, the ASIP [184] solution can also make sense because, even if off-the-shelf products are available, a deeper first investment will pay-off for the following versions.

In the third example, we will quantitatively identify when the choice of an ASIP is motivated. This will be especially true when any off-the-shelf commercial solution provides enough performance to satisfy our requirements and an ASIC brings (or can bring) more performance but incremental engineering is expensive, slow and filled with delays. Moreover, an ASIC can not keep up with changing market conditions and can not adjust to new standards that might be needed for the following versions of the product. The ASIP requires more development time in the first version of the product, since hardware, software and compiler must be developed. After that, any upgrade or change in the product just needs a compilation of the new application. But that qualitative set of statements above, can now be quantitatively supported based on the contributions that are made available in this thesis. In particular, an ASIP solution is required when the obtained performance of the DSP, namely TMS320DM644 is 51MOPS/mW , e.g. when the image size becomes too large. In that case, the energy performance ratio is improved to 361MOPS/mW with the ASIP. Moreover, we can in addition obtain more than 20 times energy decrease and also a 47% percentage area decrease compared with DSP solution and consequently, decreasing silicon cost and finally a benefit in the cost per chip. Obviously, even when the DSP meets the performance, that quantitatively evaluated energy reduction and the corresponding chip cost can also be an incentive already to go the ASIP in systems that are battery constrained. A widespread misunderstanding nowadays is to think in embedded FPGA as a final solution for small production. It is necessary to keep in mind that FPGA consumes more than other solutions even in

the case of a structured ASIC solution. Even if power consumption is not a negligible issue, the most limiting factor between ASIC solution and its FPGA counterpart is the 10 factor in terms of clock frequency. For that reason, if real time requirements are very tight, this solution must be totally rejected and this leads us to more aggressive solutions such as ASIC or ASIP.

For these reasons, if real time requirements are tight this solution becomes quite suboptimal. Then, more performant solutions as ASIC or ASIP have to be considered. Summarizing, we can classify the different solutions as shown in table 5.3.

ASIC(60fps)	**GOP type**	**Im. size**	**ASIP(60fps)**	
(opt. Speed)	III	595x487	(std. cells)	306x251
	IPP	349x286		330x108
	IPB	273x223		100x82
(opt. size)	III	361x295	(macro)	361x278
	IPP	265x217		147x120
	IPB	219x179		111x91

Table 5.3: Image size in pixels for ASIC and ASIP at 60fps

The ASIC solution has sense when the number of chips to produce is very high and provides the best compromise in terms of energy (an improvement more than 23 times the DSP) and area (45% with regard to the DSP). ASIP and ASIC must be the selected options when we want to come up with the state-of-the-art hardware architectures with better performance and hence to be used for top requirements as for example: real time video compression of 1080i (1920x1080) pixels images size, 60fps, I, P and B frames type or even in the future 2160p (3840×2160).

As a consequence, if we want to obtain a very high resolution (1080p), we must think in a solution that allows to increase the ILP and WLP (single thread) increasing the number of FUs working in parallel or adding several MPEGs working together (TLP multi-threading). Then the MPEG IP is seen as a PE (Process Element) of a multi-processor platform where PEs are connected to a bus or to a NoC. Since communication time is very small compared with computation time, partly due to the realized data memory transformations (with the DTSE approach), this allows every PE to be employed in a near-independent way in different image regions.

6 MPSOC and DVFS for QoS for the MPEG Encoder

Developers of next generation Multi-Processor Systems-on-a-chip (MPSoC) silicon platforms used in multimedia mobile devices should design efficient systems for diverse execution time vs. energy consumption trade-offs for a given quality of service. By exploiting Dynamic Voltage and Frequency Scaling (DVFS) techniques, we can obtain singular computational/power trade offs points and thus design energy efficient platforms. This chapter presents a high level methodology to acquire an optimal set of working points for a MPEG-4 Main Profile (MP) Video encoder implementation. This chapter also makes clear what is needed to adapt our compressor to a wireless channel, that should always work in coordination with the available channel bandwidth. This Encoder has to be adapted to the dynamism of the channel. In a way, that any part (encoder or wireless transmitter) does not have to compute more than necessary.

On the other hand, this video encoder can be observed as a Process Element PE of a bus or a NoC. Previously the design of a NoC was realized in SystemC (2006), this one can give out outcomes that permit to scale the results shown in chapter 4 and 5. We can have bigger image size making these tiles work in parallel. In the chapter, we show our fundamental development; though, due to its complexity, it is only explained how these interconnections would be achieved. Results can be obtained in future work and extend results to large set of data-flow systems.

6.1 NoC SystemC description for MPEG Tile

From the results of the chapter 5, we can foresee that if we want to scale the image size (in a different way that improving technological selection) several MPEG can compute concurrently . In this way we can scale up our design and therefore allow enlarging the size of the images to compress. This chapter explains which criteria would be used if the whole infrastructure was available. We describe the SystemC implementation of a scalable and parametric Network on Chip (NoC)[25]. The NoC, with a mesh (or possible torus) topology with wormhole routing. Each router is connected to four neighbor's routers through input and output channels. The channels of the routers have a data bus and two control signals to let a communication based on handshake. A wrapper establishes the communication between the resource and its environment, converting the messages into a format appropriate for the NoC. The information is packed to be able to circulate through the NoC and it is unpacked when it arrives to the destination. The SystemC [81] behavioral description of a system enables designers to obtain fast implementations. Besides, it allows to have parts of the system that are locally synchronized

by a clock but globally communicated by a protocol (handshake). This paradigm of synchronization is known as GALS (Globally-Asynchronous Locally Synchronous). The system domain is divided in sub-domains with different clocks, and the communication between all these sub-domains is done through events, or communication based on a protocol.

The router is based on the one proposed in SoCIN [185] and RaSoc[186]. The router is composed of five channels: L (Local), N (North), S (South), E (East) and W (West). Each one of these channels is composed by two opposite unidirectional channels, one as an input and the other as an output. Besides, each channel has two control signals (*val* and *ack*) that allows to carry out a communication with a handshake.

A process is presented for each channel. Then, with five independent processes in the same router, we can address simultaneously different information data incoming from different channels that go to different destinations. For example, one router may address data that comes from E and goes to N while data that comes from S goes to L.

However, we have to control the ordering priority if more than one process wants to write to the same port at the same time. We have implemented a round-robin priority policy that guarantees the port usage according to the order in which the blocks arrive to the router. Each input port has a counter and each output port has an internal busy signal to indicate if the port is being used. If this is the case, the counter of the input port goes increasing until the output port is free, and then the input port with the higher counter value will be the next to write to the output.

The five channels router architecture lets us to easily implement 2D orthogonal topologies as a mesh or a torus. These kinds of topologies have an simple routing, in our case routing XY, and they offer simple connections that enable a direct replication of blocks throughout the system. The system is established as a system of coordinates where each router in the network is identified by a pair of coordinates XY. Sending data to a resource means sending data to the router which contains it.

A difference from the RaSoc [186] router, is that data do not need the two signals *bop* (begin of packet) and *eop* (end of packet) because information have always size NxM. The L channel which connects the router with the resource has a wrapper (*Wrp*) that works as interface. The wrapper converts the NxM data into a packet that can circulate through the network. This conversion is done adding a header with the routing information to each NxM data block that comes in by the L port. The routing information of the header determines the path to take in the network, and the wrapper calculates it with the source and the destination coordinates. The resource connected to the router determines which will be the next resource to process the information. The coordinates of each resource are shared in a common file for all the routers. Every router has its own wrapper which generates its own routing table, because this depends on the resource type joined to the L channel.

In cases N, S, E and W, the wrapper checks the header's routing information of the incoming block and, if the attached resource is not the destination, it updates the routing information (decreasing the Xmod or Ymod fields) and addresses it to the output channel. If the router which reads the block is the destination, the wrapper will pass the NxM block data information, but not the header, to the resource connected to the

L port.

The communication between the resource and the router is store-and-forward: all the packets that together form the block are stored in the router and they are not sent until all NxM packets have been received. However, the communication between the routers is a wormhole: the packets are sent as soon as they arrive to the router. In this way, once the wrapper has created the packet with the header, and its routing information determines a wormhole routing through the XY routing algorithm. The XY routing algorithm always routes first in the X dimension and then in the Y dimension. This is a deterministic routing because packets always follow the same route for the same source-destination pair.

6.1.1 System Implementation-NoC

We have made a parametric router with the SC_HAS_PROCESS constructor of SystemC. This instruction lets us construct different kinds of router descriptions by only giving a parameter to the router constructor. Constructor parameters determine router behavior because these parameters will specify the routing table of each router. This is done in the declaration of each router in the main function of the system. Due to this parametric constructor we can automatically generate a torus network topology of KxK elements. It is realized indicating the K size in a *for* loop and the test bench of the channels is implemented according to the torus connection topology. Then we only have to connect the resources to the routers with the XY coordinates that we want.

The router has five SC_CTHREAD processes (clocked thread process), one for each input channel. SC_CTHREAD is the only SystemC process (SC_METHOD, SC_THREAD and SC_CTHREAD) which allows the use of the *wait_until()* call. This instruction lets us implement the handshaking communication and behavioral synthesis. At the starting point, with the *wait_until()* instruction, each thread of each channel is waiting for a *val* event, which means that some data is present in the channel to be read. As mentioned before, the router has five input channels and five output channels. Channels L, which connect the router with the resource, have been implemented with the primitive s*c_signal.* On the other hand, N, S, E and W channels have been implemented with *sc_fifo* channels of SystemC. In spite of the fact that the *sc_fifo* channel is not a synthesisable channel, it allows us to implement a fast simulation of the system and determine the FIFO sizes for a future synthesis of the system. *Sc_fifo* is a SystemC channel that may contain any data type and it is used to manage data flow. Consequently, it can manage the structure packet build *by* the wrapper (NxM data block and a header). For the simulation, we indicate the number of clock cycles with the sentence wait(t) (where t = NxM + header).*Sc_fifo* is a SystemC primitive that allows to generate fifos of any size, we can create its equivalent RTL synthesisable, similar to a work presented in our department in [187]. Depending on the type of simulation (high level, low level) it is possible to use an entity or another.

The packet header size is of 16 bits, where 8 bits are routing bits and the other 8 bits are reserved for future information. From the header's routing bits Xdir and Ydir have a 1-bit size, and Xmod and Ymod have a 3-bit size. The 3 bits of Xmod and Ymod

fields let us implement NoCs of 8x8 routers that means the possibility to connect until 64 resources. However, the tests we have done have been performed on a network of only 3x3 elements.

6.1.2 Routing Simple in a MPEG Compressor System and Environment Description

In this section, we show an example of communication between resources of a MPEG video compressor with handshake communication and NxM (normally N = M = 8 or 16) pixel data block. The images to process can be of type I, B or P, and these image types determine the data flow in the system. In reference [94] is explained in detail the MPEG compressor functionalities. When a resource sends the information to the router, the wrapper builds a packet structure with a header and NxM pixel data block. The image type determines the block image size, which can be bigger than NxM. Consequently, according to the image type, it may be necessary to do P transitions of packet structures with image blocks of NxM sizes. For example, the basic block image of NxM corresponds to 8x8 block images. The transmission of an I-image will require the next steps: first to packet the information (8x8). Second, to indicate in the header that it is an I-image and the routing information. Finally, it will be sent to the next router according to the routing information.

However, in cases of B-images and P-images, the blocks to send are of 2Px2P (P = search region from 16 to 64). If we take again our case as example, where P value is 16, we need to send 16 blocks of 8x8 pixel size to completely send a block image of 2Px2P (32x32). Each one of these 16 blocks is packed by the wrapper with the correspond header. All 16 packet headers will have the same routing information because they belong to the same image, which goes to the same destination resource [94, 188]. The wrapper, according to the image type and to the resource connected, calculates the routing information to write in the headers. The routing information depends on the pairsof source-destination coordinates. General routing tables for the different image types and simulations are presented in extended manuscript in [189, 190]. Figure 6.1 is a diagram where these ideas are shown.

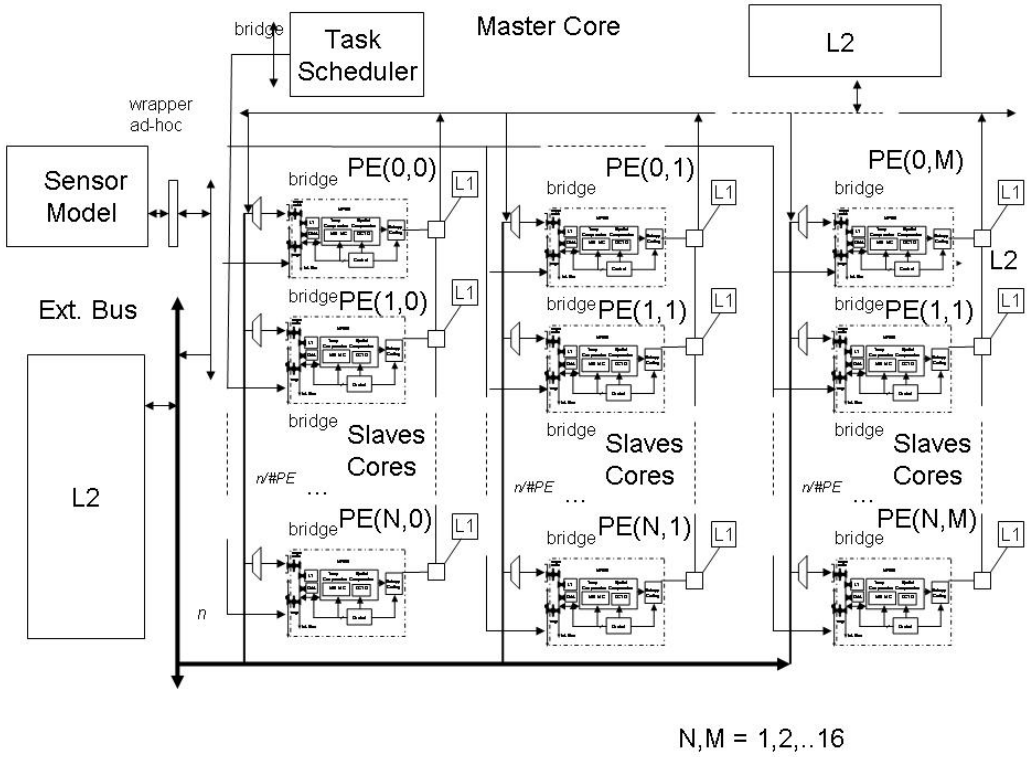


Figure 6.1: Architecture representing the PE-MPEG connected to a dedicated NoC

6.1.3 High Level NoC Simulation

As already explained, a scalable and parametric NoC has been realized in SystemC. Each router that is part of the NoC has its own routing table according to the resource connected. The generation of the NoC with mesh-torus topology is automatic; to generate it, we only have to indicate the size. In our tests, although the network topology can represent either a torus or a mesh, the routing algorithm has been implemented only for a mesh network. The routing algorithm for a mesh network has the advantage that the routing tables are smaller and the disadvantage that the latency is higher, because in some cases we will cross more routers than required. However, because a network of only 3x3 routers is quite small, we prefer to simplify the size of the tables to reduce latency. The system has been verified with the simulation of an image compressor system, where an I-image compression has been executed. As we have mentioned in section 6.1.1 (Implementation in SystemC) the *sc_fifo*channels are not synthesisable. Consequently, at the moment, only a part of the router is synthesisable. More work can be done with the ideas extracted from [187]. However, once that the system has been verified, we can refine it to obtain a new synthesisable version. We leave this part as a future work, where we will try to synthesize the full router. In chapter 7, we explain the experiments

that might be realized if we had a NoC synthesized in a known technology.

6.2 Multidimensional HW-SW Implementations

In multi-criteria optimization or decision problems there are several objectives potentially reliant on contradiction [191]. According to the relative weight of the objectives, diverse solutions may be chosen. The concept of Pareto optimality, identified by Vilfredo Pareto who introduced it in [192] as 'maximum ophelimity' in an economic system, affirms that a solution is optimal if it is unfeasible to find a solution which improves on one or more of the objectives without damaging any of them. If one result is enhanced in one objective than another solution and not worse in any other objectives, the latter is dominated by the former, and should always be preferred. Surveying the likely solutions to an optimization problem then amounts to finding all solutions that are not dominated by any other (feasible) solution. This set of solutions is called the Pareto frontier and is guaranteed to contain all optimal solutions, whatever way the individual objectives are weighted. In other words, the Pareto frontier precisely detains the available trade-offs between the diverse objectives.

Multi-criteria optimization problems emerge often in a broad range of fields including finances, economics, business, and all kinds of engineering-service management in micro networks of embedded resource-constrained devices. Choosing an optimal configuration for a dynamic constellation according to the available resources requires optimization of power consumption, quality, timelines, and so forth, at run-time. For instance, the available configurations of the likely sets of points may be obtained from off-line Pareto analysis of different architecture mappings. Some configurations may be enhanced for efficient energy utilization, others for getting an upper quality, and so forth. The selection of a configuration at run-time requires composition of Pareto-optimal configurations of parts into Pareto-optimal configurations of a system. Some of these operations will have to be performed at design-time on resource-constrained devices model. The number of configurations has to be kept small and partial selection among optimal configurations may be needed to keep the number relatively small.

6.2.1 Pareto Points Related Work for QoS

Pareto analysis is employed in a lot of engineering applications [191, 193, 194]. In our own field of research for instance, we can point out its use in design-space exploration [195, 196], application mapping on embedded multiprocessor systems [197, 196, 198] or architecture exploration for Systems-on-Chip (SoCs) [199]. There is also a huge effort on Pareto analysis and multi-objective optimization and decision making in general (see, e.g., [200] for an overview). This kind of work traditionally focuses on characterization of the Pareto frontier based on the nature of the cost (objective) functions (typically continuous functions on real numbers), algorithms for computing the Pareto frontier or approximations of the frontier [201, 202]. As far as we know, this is the first time that there is a study of different trade-offs in terms of execution-time and energy consumption according to the number of functional units in the platform. According to my knowledge,

studies of video compression just had into account Quality/Bit rate without taking into account that for embedded systems energy is of primary importance. Algorithms to compute the Pareto frontier include genetic or evolutionary algorithms [203, 204, 202], and tabu search algorithms [205, 206, 203, 200, 207, 191, 199, 208]. Algorithms to identify Pareto points in sets of configurations (also called Pareto minimization) are introduced, although according to [193], literature on identifying Pareto sets is sparse. The algorithm of Bentley [206] is asymptotically very efficient. [193] introduces a hybrid algorithm which combines it with a straightforward algorithm which is efficient for small sets. Data structures for storing sets of configurations have also been studied [209, 197, 196, 198, 193].

In this section, we briefly discuss some engineering applications, and in particular dynamic Quality-of-Service management in micro networks of embedded resource-constrained devices. Choosing an optimal configuration for a dynamic constellation according to the available resources requires optimization of power consumption, quality, timeliness, and so forth, at run-time. The available configurations of likely sets of points may be obtained from off-line Pareto analysis of different architecture mappings, for instance. Some configurations may be enhanced for efficient energy utilization, others for getting an upper quality, and so forth. The selection of a configuration at run-time requires composition of Pareto-optimal configurations of parts into Pareto-optimal configurations of a system. Some of these operations will have to be performed at design-time on a resource-constrained devices model. The number of configurations has to be kept small and partial selection among optimal configurations may be needed to keep the number relatively small. A complete survey about how to use Pareto points for embedded systems can be found in [210].

6.3 Dynamic and Voltage Frequency Scaling: DVFS

These techniques of power management in computer architecture serve to manage the power consumption depending on when the device is connected or not. It is based on the use of voltage scaling and the frequency scaling. These techniques allow to carry out a task in a faster or slower way managing the frequency (device speed) and the voltage (time of answer of the transistors' gates).

6.3.1 Dynamic voltage scaling

Dynamic voltage scaling is a power management technique in computer architecture, where the voltage used in a component or part of it is increased or decreased, depending upon circumstances. Dynamic voltage scaling to increase voltage is known as overvolting; Dynamic voltage scaling to decrease voltage is known as undervolting. Undervolting is done in order to conserve power, particularly in mobile devices, where energy comes from a battery and thus is limited. Overvolting is done in order to increase computer performance. CMOS-based digital circuits operate using voltages at circuit nodes to represent logical state. The voltage at these nodes switches between a high voltage and a low voltage during normal operation—when the inputs to a logic gate transition, the

transistors making up that gate may toggle the gate's output. Each node in a circuit has a certain amount of capacitance. Capacitance can be considered as a measure of how long it takes to effect a given voltage change for a given current . The capacitance arises from various sources, mainly transistors (primarily gate capacitance and diffusion capacitance) and wires (coupling capacitance). Toggling a voltage at a circuit node requires charging or discharging the capacitance at that node; since currents are related to voltage, the time it takes depends on the voltage applied. By applying a higher voltage to the devices in a circuit, the capacitances are charged and discharged more quickly, resulting in faster operation of the circuit and allowing for higher frequency operation. The dynamic power switching power dissipated by a chip using static CMOS gates is:

$$Power\_switching\_=S\ C\cdot V\hat{}2\cdot a\cdot f \tag{6.1}$$

where C is the capacitance being switched per clock cycle, V is voltage, and f is the switching frequency, therefore this fraction of the power consumption diminishes quadratically with voltage. Moreover, there is also a static leakage, the current leakage, which has become more and more accentuated as feature sizes have become smaller (below 90 nanometers) and threshold levels lower. In state-of-the-art deep sub-micrometer technologies, in 2008, dynamic power accounts for approximately two-thirds of the total chip power, which limits the effectiveness of frequency scaling.

Thus, dynamic voltage scaling is commonly employed as part of approaches to manage switching power consumption in battery powered devices such as cell phones and laptop computers. Low voltage modes are used in conjunction with lowered clock frequencies to minimize power consumption associated with components such as CPUs and DSPs; only when significant computational power is needed will the voltage and frequency be raised. When current leakage is a significant factor in terms of power consumption, chips are often designed so that portions of them can be powered completely off. This is not usually viewed as being dynamic voltage scaling, because it is not transparent to software. When sections of chips can be turned off, as for example on TI OMAP3 [211] processors, drivers and other support software need to support that. The speed at which a digital circuit can switch states - that is, to go from "low" (VSS) to "high" (VDD) or vice versa - is relative to the voltage differential in that circuit. Reducing the voltage means that circuits switch slower, reducing the maximum frequency at which that circuit can run. This, in turn, reduces the rate at which program instructions can be issued and may increase run time for program segments which are sufficiently CPU-bound.

This again highlights why dynamic voltage scaling is generally done in conjunction with dynamic frequency scaling, at least for GPP. There are complex trade offs to consider, which depend on the particular system, the load presented to it, and power management goals. When quick responses are needed, clocks and voltages might be raised together. Otherwise, they may both be kept low in order to maximize battery life.

6.3.2 Dynamic frequency scaling

Dynamic frequency scaling is another power conservation technique that works on the same principles as dynamic voltage scaling. Both dynamic voltage scaling and dynamic

frequency scaling can be used to prevent computer system overheating, which can result in program or operating system crashes, and possibly hardware damage. Reducing the voltage supplied to the CPU below the manufacturer's recommended minimum setting can result in system instability.

Temperature: The efficiency of some electrical components, such as voltage regulators, decreases with increasing temperature, so the power used may increase with temperature. Since increasing power use may increase the temperature, increases in voltage or frequency may also increase system performance. Dynamic frequency scaling (also known as CPU throttling) is a technique in computer architecture where a processor is running at a less-than-maximum frequency in order to conserve power. Dynamic frequency scaling is commonly used in mobile devices and sensor networks, where energy comes from a battery and thus is limited. It is also used in quiet computing settings and to decrease energy and cooling costs for lightly loaded machines. Less heat output, in turn, allows the system cooling fans to be throttled down or turned off, further decreasing power consumption. It is also used for reducing heat in badly cooled systems when the temperature reaches a certain threshold. Most systems affected by this are inadequately cooled overclocked systems.

6.3.3 Switching Capacity

In tables 5.1 and 5.2 of the previous one we can observe that chip energy is, for a large part, due to data memories and data registers for most applications. For the target technology of 90 nm, the leakage current is still not dominant though. We remind that for the ASIC memories the dynamic energy is given for the worst case, i.e. This is not the case at all for normal application usage patterns.

In the papers available in the literature no study is presented on how this switching capacity affects the total system capacity. That is to say, according to the information that we are storing in the memories, the required capacity can be significantly higher or lower. The extreme situations would occur for the case under study (MPEG), on the one hand, for an image of just a colour. Then, the stored values (pixels) will be all equal and the calculations a priory will be mostly deactivated so the actual switching power would be (much) lower.

On the other hand, we have the case where we are compressing images with a lot of entropy, that is to say, much variation exists inside the image. Then, the many stored values make the transistors switching capacity become (much) higher. The extreme case is that we are compressing an image of white noise. This dimension (power consumed) is not orthogonal (independent) from the image quality. Indeed, to obtain a quality of similar image sometimes the computation must be higher and sometimes lower. It would be interesting to carry out a study with different image types to detect how they affect the power with the image quality and the bit rate. This would also affect the chip temperature because a higher activity due to the images would promote an elevation in the temperature. We can define *node activity* according to the definition 6.

Definition 6. *Given the register A, Ai a bit i of A and after a clock cycle ζ we get A'. We define D as:(A Λ A ') and if we define the activity k=0 initially and for the whole*

bits of D (∀Di&1 == true, k++) . Ultimately k it gives us the activity of this register in a ζ cycle.

This may be realized by monitoring the switching capacity counting toggling in the memories and in other elements storage (e.g. power hungry data-paths). The code below computes this activity and it would be easily translated to VHDL to make a hardware synthesis. It would contribute a small subsystem added to the different nodes of the devices. This code represents a program that computes the activity of a register (A) or memory locations. Based on the results, we can then come up with representative information on how the dynamic energy for the encoder realization and the image quality are related[212]. This activity computation must be placed in memories and registers where different operations with pixels are carried out. This would allow us to estimate what kind of image (entropy) we are compressing and advance in the system adaptation in order to compute with the suitable resources.

```
#include "activity.h"
        double activity_ (int &A, int &Aprima ){
        int ccc = A^Aprima;
        int value = 0;
        double activity =0;
        for(int o=0;o<8*sizeof(int);o++){
                bool k= 0;
                k = ccc &1;
                if(k) value++;
                ccc=ccc>>1;
                cout<<value<<" ";
        }//for_o
}//activity
```

Figure 6.2: Activity computation in node A

6.3.4 Coder Code Transformations

The initial point of the flow is a C++ specification of the MPEG-4 standard. From this specification, we develop a MPEG-4 Main Profile (MP) code without dynamic memory allocation and without pointers. This sort of code is required for final SystemC behavioral synthesis (as already explained in chapters 4 and 5). Afterwards, on this refined C++ code, we carry out a series of platform independent data structure transformations (DTSE [153, 154]) to obtain code that minimizes memory transfers between Level 2 memory (L2) an the internal cache (L1). Platform dependent transformations optimize the size of the data structures for an efficient fit on the specific target memory architecture.

Later on, we modeled a SystemC description of the resulting C++ code and we analyzed the performance and gain obtained by applying Dynamic Voltage and Frequency Scaling (DVFS) [213] for multiple clock domains (frequencies). As the result of the analysis, we can derive dissimilar performance-energy efficient working points. A scheme about the code transformations is shown in fig. 6.3.

Our SystemC model, as already mentioned, is partitioned in two parts: data flow and control flow dominant part. The data flow dominant part is mostly responsible of the transformation of each pixel in every macro-block and it is responsible for 80% of the computation. Data flow part of the encoder is a good target to exploit any available Data-Level (DLP) or Instruction-Level parallelisms (ILP). For our experiments, we studied the algorithms required to compress a macro block. Macro block can be: I (intra), P (predicted) or B (bidirectional). As explained in chapter 2, the I blocks are just spatially compressed with the two dimensional DCT algorithm. P macro-block temporally compresses with a forward ME and also a spatial compression. B macro-block needs two ME (forward and backward vectors) and a spatial compression. Hence, I macro-blocks are the less computational intensive and B ones have the higher requirements.

6.3.5 Results

A series of experiments were done to obtain different Pareto points. A more precise explanation is offered in appendix A. We have added the results in the appendix because they have obtained very high level and they were realized by an older technology (130nm).

To obtain optimal points in a HW platform, it is necessary to have information (energy, execution time) about the basic elements that form our model (how it is built): Memories, registers, elements MAC, adders, shifts, etc. This information is acquired at low level (target technology) and it is written in our SystemC platform model in high level to obtain Pareto points. These points were changing depending on the number of FUs(MACs blocks) that are in use and the type of macro-block to be compressed. Taking as reference the high level results shown in figure 7.6, we can analyze the real performance in Crisp-Trimaran target platform changing the number of functional units for a fixed voltage and frequency environment. That specied architecture can cope with up to 12 functional units. To execute the same application there is a range of work-points and not only one, as when engineers design for the worst case scenario. In a similar way, Pareto points have been obtained for an ISP. In this case, there is a XML representation of the platform with its technological values (energy, execution time). Figure 7.7 (Appendix) illustrates the outcome of this study. Therefore, depending on the number of FUs working in parallel and the type of image we can obtain a curve with the optimal power values. Also, the same application has been mapped into five different platforms which show diverse hardware configurations and the time to accomplish the encoding of I, P and B macro blocks. New multimedia applications have recently increased in complexity and demand high performance at low power. Next generation of portable devices will present different energy-aware work points to optimize battery life. The previous results show that the OWP are not always easy to find. Different considerations as frequency and voltage (hence consumption) and the number of functional units can not be neglected if we want to maximize battery life. Run time dynamic voltage and frequency scaling will allow us to save a lot of power. More effort has to be done for obtaining better quality-bit rates space for our application, so, we have to characterize MPEG model with more input standard images (at this moment we have

just used foreman). More information about this work is shown in APPENDIX A and references [214, 215].

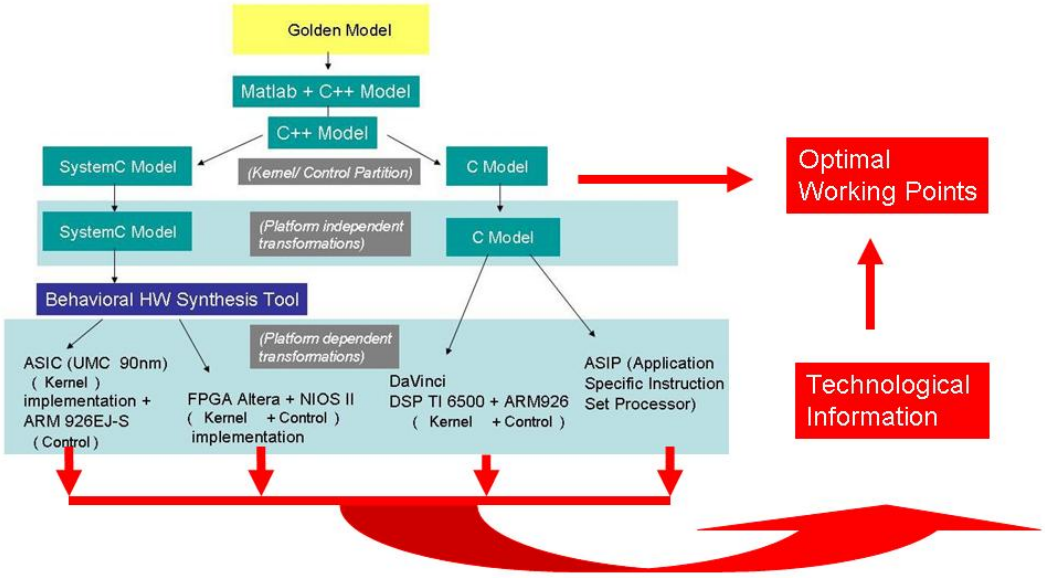


Figure 6.3: Design flow related to code transformation

7 Conclusions and Future Work

The main goal of this thesis is to obtain a set of results for the implementation of a given system level application down to different architectural platforms. This allowed to carry out a fair comparison that includes to build the whole system and to complete the design chain to the diverse silicon targets. This comparison uses four variables for its evaluation (execution time, chip area, energy consumption and design time) and produces a map of different optimal implementation points according to a given set or operating requirements. I built a complete MPEG-4 MP. This standard is a well known reference example, pretty popular in the scientific literature and this compressor is also a fine example of data-flow application. Therefore, results extracted from this thesis can be extended to other data-flow applications. I considered necessary to compute image compression with real-time constraints. Hence, I would like to dispose of the most flexible design possible in order to map the same specification into the different platforms.

For that purpose, I chose SystemC/C++ as description system level language and set-up the different implementation flows for the different architectural and silicon platforms. This powerful framework allows comparing implementations in a reasonably objective way. Since our results come from a unique reference model and all designs were finally mapped in the same silicon technology (90nm CMOS).

The result of this research work is a set of criteria and a map of the available solutions on the performance space rather than an assertion saying that a unique solution is better than others. My intention has been to develop techniques and formulate methods that increased design productivity. This development can be further applied to the new parading of implementations: those that use DVFS techniques and NoC-based MPSoc implementation explorations 7.3.

7.1 Main contributions

Thesis contributions can be divided in 2 types: quantitative and qualitative contributions:

We consider the most significantquantitative contribution is the development of the model able to achieve the different experiments: the MPEG compressor that has been realized in SystemC/C ++. It is designed in a way that multiple implementations are possible, ranging from a large part in HW up to loaded in an accelerator as a VLIW . In case of the FPGA and ASIC, two implementations have been carried out. We obtained a set of values for seven different implementations targeting four different HW platforms (FPGA, ASIC, DSP and ASIP) with diverse internal architectures, selected to get optimal points. In the case of ASIC, we managed to end up with the layouts of

the two solutions. This led to an increase in efficiency of 56 % for speed versus 26 % for energy (in FSME solution 20% for speed and 57% for energy in FAST solution). In case of the ISPs, code improvements have been accomplished to come up to more ideal solutions with regard to those who would be obtained by a direct implementation. In case of the ASIP the improvements have not only been realized in the code but also in the silicon microarchitecture that form the VLIW.Other qualitative contribution is the accomplishment of a functional NoC in SystemC.

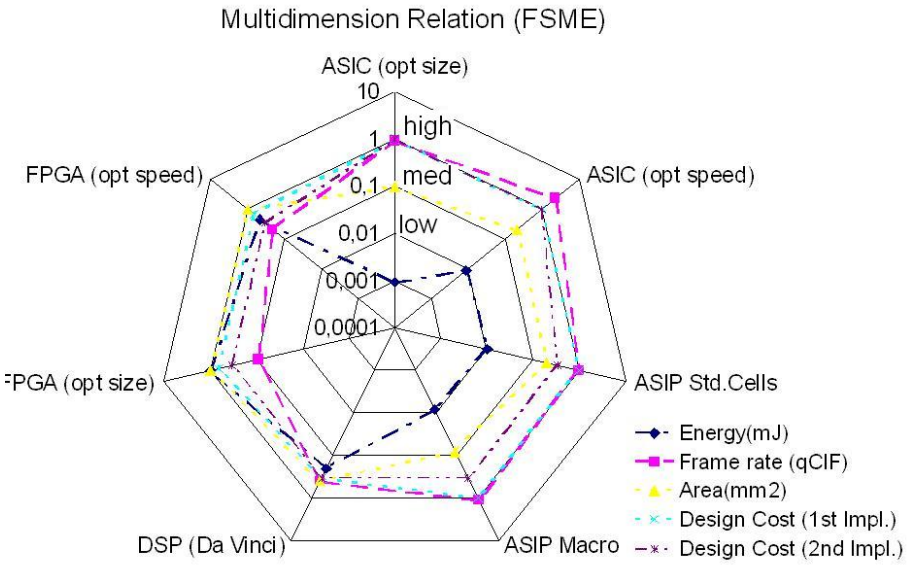


Figure 7.1: Multidimension Relation for MPEG with FSME algorithm.

The summary of the quantitative contributions is presented in tables' 5.2 and 5.1 where they provide absolute values of the 4 variables under study. A summary is shown in figure 7.1. It represents all the different platforms and the relative values among them. The qualitative contributions can be included in the dissertations of the paragraph5.6, where I explain when it is more suitable an implementation or another. Also in chapter 6 we can see as qualitative contributions the possibility of considering the MPEG as a PE connected to a bus or a NoC. We also clarify that every macro-block can be controlled with different requirements, partly owned by the DVFS and therefore there is a great dynamism; this is explained in the appendix and how it could be treated if we consider a multiple PE.

One of the main interests of our group has been the evaluation of platform based and model based design methodologies,and we have published the following papers [65, 216, 217, 218, 219, 220]. Overall, we were interested in the system description for digital image processing in real time and / or cameras based on hardware platforms where the prototype device was an FPGA [221, 66, 217, 67].

During my stage at IMEC (Belgium) I distinguished the possibility of exploiting co-design with a language named OCAPI-xl; this language, based in C for HW, was similar

with the posterior language called SystemC. I used this system language for my systems descriptions and build the system descriptions we have published [222, 223, 224, 225, 226, 227, 189, 228, 229].

We have made some effort in selecting the most appropriate language for co-design system development, to be adapted by our group in order to describe the heterogeneous systems in a HW/SW approach. With these ideas we published [226, 227].

Hence, after some discussions I based my system depictions in SystemC Language [230, 228, 229]. As well, I was interested in observing the compressor as a PE connected to a NoC or a bus for scaling results. Then different compressor works in different parameters and aided with Ramon Pla's interest we could develop a NoC in SystemC that we connected to our compressor [190, 189]. Furthermore, with the interest of the Dr. Lluís Ribas, we published an FDL paper that was a work based on micro-architectures described in SystemC [231]. Then, I had a complete stable version of the image compressor with the possibility of I, P, B frame compression. With the effort of Guillermo Talavera, we made a study based on DVFS, and published the following papers [232, 215, 214]. Furthermore, I have contributed in the design of an environment for data acquisition and in a frame work for SW-SystemC co-simulation [233, 234].

In chapters 3 and 4 we have realized with IMEC collaboration a comparative study of several hardware platforms (ASIC, FPGA, ASIP and DSP) where a relatively complex algorithm has been implemented since it is an MPEG-4 Main Profile image compressor, which allows slices I, P, B macro-block compression with a configuration 4:2:0. As already mentioned, one of the major contributions of this thesis has been to describe a complex system like it is an MPEG-4 compressor in a Co-design language like SystemC. This IP has been optimized for different platforms and we have obtained metrics that allow us to lucubrate what platform is better at everymoment. These results can be extended to any system based on data-flow and that has low power consumption requirements because it must be connected to a battery. The study has been realized comparing each platform from a raw implementation to a most optimal implementation; and also comparing the results among the different design platforms providing real numbers and relative values among platforms. This study permitted us to submit two journals papers[235] and [236] that are under revision.

As a teacher I also have achieved some educational publications like [237, 238, 239, 240, 241, 242, 243, 244].

7.2 Impact Analysis

As a future work, it would be interesting to make a deeper effort than in [215, 214], where a study of the power vs. execution time was realized for different FUs. It would be useful to extend for other parameters that users demand (QoS)–i.e. obtaining different Pareto points for different curves that involve bit-rates, and video qualities. And depending on the energy that the images and the QoS required by user to take different alternatives of compression. Till now there are very few works that study the modification of an electronic system depending on input data-pixel in a totally un-deterministic way. As a

future work we can also realize QoS's study for what users expect of a video on demand. Different parameters related to the QoS (bit-rate, Quality, execution time, and energy) could be modified for the image compression. Though in reference [245] a related work has been realized., it is based on a wireless transmission detailing that the compression is already carried out with different layers and requirements. Then compressed information must be transmitted by a wireless channel. Hence, it has been based on a collection of compressed videos that had to be transmitted wirelessly with different requirements. Instead of making it this way, a complementary method would be by obtaining metrics with the compressor that has been built.

The first part of her work focuses on a WLAN model, which has less mobility and comparatively less channel variations than a 3G channel. In this part of work, this thesis first demonstrates the effectiveness of introducing the application awareness into a PHY-MAC optimized framework [246]. This thesis then introduces the video quality into the design-time phase. The solutions proposed in [247] schedule the packets by both exploiting link layer scaling and sleeping trade-off. In [248], it exploits the temporal scalability brought by the prediction dependencies among video frames. An extension to this work would be to focus in optimizing the compressor. To do that, we can extent the compressor to make compression adapt the bit-rate to the wireless link. In our work we have compressed always for the worst case to get a maximum PSNR; it is also possible to extend the work adapting the bit-rate to the wireless channel and optimizing resources and / or energy.

In chapter 4 and 5 we have been shown that a single MPEG can manage compression images of size 595x487@60fps with I frames. This image size can be scaled using different MPEGs IP working in parallel. Thiswas one of the motivations to use a bus or a NoC connecting several PEs. Results could be extended to PE that are based on data-flow systems. Here, the future work could be related with previous works [249, 250]. With information obtained from chapter 4 and 5, tables 5.1 and 5.2 regarding the different properties of macro blocks types compression (max. frequency, execution time, energy consumption, PE area). We can expect the number of slaves (PE) which can be connected to the bus and /or NoC.

When the number of slaves is high, data can be deteriorated due to the delay in the bus. Although, in parallel computers using shared bus we can achieve up to 70 processors [251]. Network-on-Chip (NoC) topologies such as mesh or fat-tree utilize shorter wires and offer substantially larger maximum bandwidth but also higher latency with small load [252, 253]. The runtime is the sum of computation, communication and synchronization. More precisely, only the non-overlapped portions of each in the critical path are considered. The future paper would be an extension of the paper on the impact of communication on the scalability of Data-parallel Video Encoder on MPSoC [250].

In our point of view, this paper [249] can be extended because they only take into account the input data video transfers, since they account approximately 99 per cent of all the traffic. This is true if all encoding is measured during INTER frames. But there is not a scenario where there are just inter frames. The intra frames must also be taken into account, since they have primary information essential to reconstruct the frames. Then the relation between input data and data compressed (traffic) is not 1/100. We

mean the traffic after data compression is not negligible in a real environment. They neither show a real NoC with real communication pattern to know in an accurate method the cycles needed to make data transaction to the diverse PEs.

In reference [254] we observe that the higher power consumption is and the longer bus width, the lower the numberof inputs/outputs ports , and that the consumption decreases at the cost of an increase in area. In our case it is visible that a Spatial Division Multiplexing (SDM) solution is a technology that exploits the fact that network on-chip links are physically made of a set of wires. SDM consists of allocating only to subset of the link wires to a given virtual circuit.

If data pixel storage has been realized in a pertinent mode we can pass from N bits's bus that connects the memory via a de-multiplexer to N /#PE bits that connects each tile (PE). In this form, we have a very data intensive (data flow) application, where each tile can work in a nearly independent form in a divide and conquer technique (i.e. 256 bits connection to 8 PE with a L1 memory) . Each PE bus has 32 bits. This bus size permits 4 pixels per transition. The number of PE must be given by the existing relation between macro-block computation time with regard to the communication time.

It must be evaluated if this solution even being very intensive communicating the different PE is a correct option. It must also not be very power hungry. Therefore, it should be experimented the relation between bus memory and bus that connects the diverse PE. It must be estimated the maximum number of PE connected. It must also be estimated the number of PE connected to scale the image size for our fps requirements. It should be evaluated if applying also a Time Division Multiplexing (TDM) makes sense. As a first approximation, the proposal would be to connect the input PE bus with the SDM technique and the output bus PE set in a Time Division Multiplexing (TDM) fashion. The output data compressed is loaded in another L2 memory. View figure 6.1.

The maximum PE output bus communication priority must be low power because the quantity of information is quite lower than the communication in the input PE. In this case 2 possibilities exist a) fulfill a wormhole with a store-and-forward with the smallest possible flits; or b) fulfill a store-and-forward of the compressed macro-block. If this second option is attained, there is the possibility of realizing a packages reordering, if it was necessary. This would be carrying out when there is very much dynamism to compress a macro-block. Then it can happen that a block ends before another. Therefore, it can be presented a packages reordering while they travel by the different NoC tiles, or when compressed information is written in the L2 memory where they have to be written in the proper place. For example figure shows that the order of finishing each PE is different, this is because it is possible that each PE works with different pareto (energy, execution time, bit-rate, quality) requirements.

7.2.1 Application-specific topology: Scenario Definition

It is clear that every Domain of applications need a specific type of communication system. In our case, for scaling the MPEG results, the communication topology has to be specific and very related with the application. This communication could be extended

to data-dominant applications. We have the images in external memories as for example in DIMMS with eight DDR3 [255] chips (that they have a bandwidth 12800Mb/s to 25600Mb/s each). We can have several DIMMS storing the GOP. The most important of this NoC must be the use of the maximum throughput with minimal latency and applying race to idle (we mean, to run clock to the maximum frequency the minimum possible time to be able to preserve power consumption). And then, if it would be crucial to turn off regions with no computation. On the other hand, we tried to develop a NoC where every communication node would be employed at different optimal working points; that would make it too complicated and we think that it is not tailored to very dynamic PE, since our tiles of computation are quite homogeneous (just a kind of PE).

The relation between computation and communication is different depending on the macro block type to be compressed (I, P or B). And the type of PE that we have already developed, that may be an FPGA, an ASIC, a DSP, or an ASIP. This relation allows us to know the number of PE that can work in parallel. For example in the hardware platforms (FPGA and ASIC) with a bus of 32bits, the number of communication transitions with regard to those of computation are for I (computation vs. communication: 10,4), for P (7,2) and for B (16,36) if they are the optimized for speed, and d:16 research region of P and B images. See figure 7.2a). In the case a) is when the whole macro-block is necessary to start computation, but this is not completely true. For example for one image of the type I, it is possible to begin to compute when the first block is loaded. This can be carried out making a loop morphing of the loops that they operate to load the information with regard to the loop that starts computation. This not only allows start computing a *delta* time before. In addition, we can calculate more elements in parallel (more than 40 for Image I type). See figure 7.2 b). Furthermore, if d:64 for example, for a sequence with a lot of dynamism a basketball match for example the relation in P and B frames would increase the number of PE to more than 64.

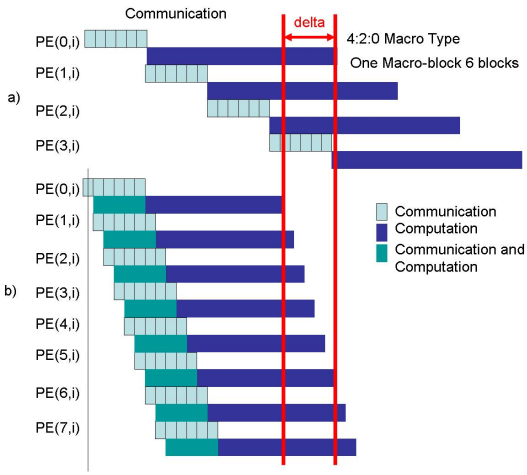


Figure 7.2: Communication vs. Computation for I frames

These relations allow us to know the number of PE (for d:16) that goes from 7 to

40. The information that has to receive every PE is well determined at every moment and it is understood that the DMA gathers the data structures (pixels information) in the proper way from the external memories. One of the most important issues is that every PE is always computing. Therefore, the bandwidth between the external memory and the PE has to be the suitable, since the computation must be very data intensive. The proposal is that the number of bits that connect the L2 memory would be high for example 256 bits (one DIMM with eight DDR3 chips, each DDR3 provides 64bits of information), this bus distributes the signals in sets of 32 bits that connects every PE. Thus, every PE can receive 4 pixels in parallel (4 bytes in every transmission). For this solution if the number of (#PE) goes from 7 to 40, it has sense to use a bus connected to the different PE, ss shown in the figure 7.3. In the figure 7.3 we observe that a different PE is connected to a dedicated bus and there is an SDM (spatial division multiplexing) communicacion. We denote, since it is an application data-dominant, that information arrives to every PE as fast as possible. For this reason, many bus lines are used in parallel to connect the different PEs. Once the information has been computed the output communication requirements are minor, since the amount of information to transmit has diminished considerably. It is in this case output bus must have fewer bits than the input bus.

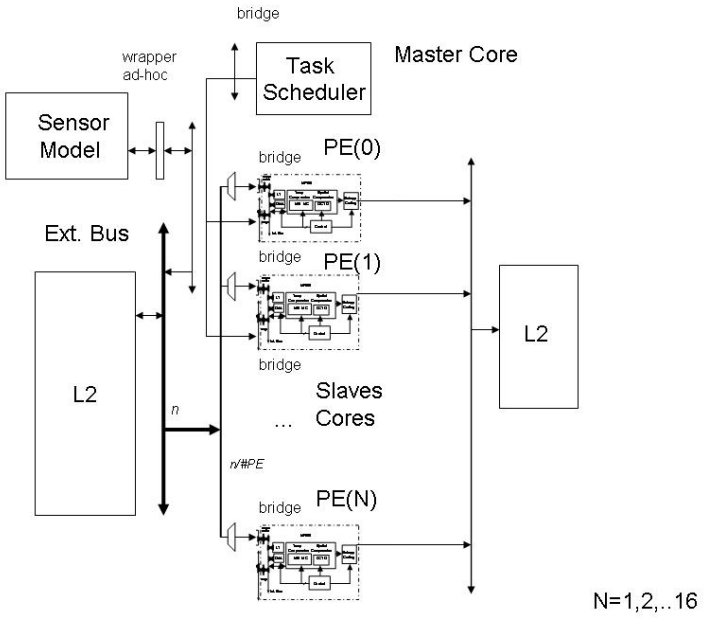


Figure 7.3: Several PE connected to two dedicated buses

The experiments become more interesting if we add dynamism to this scenario. For example, we know there is a Worst case (maximum time to compute a macro block), but that there is a range of values for processing the macro-blocks (see Appendix A).

Therefore, every PE can process the macro-block with different requirements of energy, execution time for example, with an image constant quality. Or in a similar form the desired bit-rate; this can we related with the thesis [245] . For example, by controlling the bit-rate is possible to send the maximum data through a wireless channel.

7.2.2 Scenario Example

For example we have the NoC structure as the one that is observed in figure 6.1. Where 64 PE are connected to a NoC. Each PE can be working at a different optimal point. This can be because every PE can have dissimilar requirements. In figure 7.4are observed 3 PEs (PE (0,0), PE (1,0) and PE (2,0)); each PE has a different working point to obtain a similar quality. In a similar way, it is feasible that for example PE (1,2) ends before than PE (0,2). We mean it is possible to produce information races or hazards. This would produce that although all PE row begins in similar time. The information is computed with different times and therefore this must be reordered. This reordering can be managed while it travels by the NoC or when compressed macro-block is saved on the L2 output memory. It is essential to bear in mind that compressed macro-block size is totally heterogeneous and therefore, it is not possible to enable a constant memory size in L2 output memory.

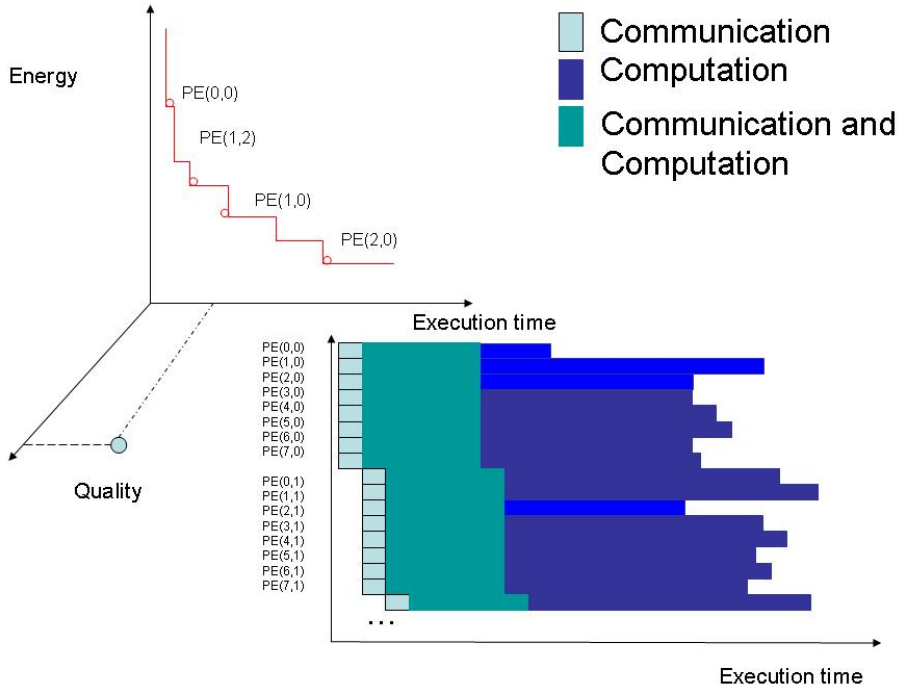


Figure 7.4: PEs Dynamic Computation produces information races

In section 6.3.3 from chapter 6 is commented that energy can diminish when input information is taken into account, we have to take into account activity (data information entropy) that is estimated in certain nodes. Applying this methodology is possible to estimate power and then temperature. Besides, it is clear that work-load always can increase producing a warm up in a PE. This can produce in long term a malfunction of the node due to an overheating. It would be useful and feasible to switch to another node that has been used in short period of time. In reference [210]this idea has been described partly as a part of a more complex tool-flow where this part is defined as system calibration (defined as: calibration mechanism, which allows the system to learn on-the-fly during its execution, and to adapt itself to the current input stimuli, by extending the scenario) for System-Scenario-Based Design of Dynamic Embedded Systems.

7.3 Future Work

In future there will be thousands of million transistors in a chip. It is very feasible that some of these transistors will not work or with the time they will stop working and therefore, the system will have to be redundant. These possible failures will ought to be calibrated to realize the different tasks with the QoS of the applications with experienced malfunctions.

According to section 7.2 is proposed three future lines of research. First, thesis [245] can be extended in the compressor side making the compression an adaptation to bit-rate of the wireless link. In our work we have compressed always for the worst case to get a maximum PSNR, or it is possible to extend the work adapting the bit-rate to the wireless channel and optimizing resources and / or energy.

Second, in chapter 4 and 5 has been shown that a single MPEG can manage compression images of size 595x487@60fps I frames. This image size can be scaled using different MPEGs IP working in parallel. This was one of the motivations to use a bus or a NoC connecting several PEs. Results could be extended to PE that is based on data-flow systems. Here, the future work could be related with previous works [249, 250]. With information obtained from chapter 4 and 5, tables 5.1 and 5.2 regarding the different properties of macro blocks types compression (max. frequency, execution time, energy consumption, PE area). We can predict the number of slaves (PE) which can be connected to the bus and /or NoC.

Third, if each PE has different working point to obtain a similar quality. The information is computed with different time's requirements. Hence, output PE data must

be reordered. This reordering can be managed while it travels by the NoC or when compressed macro-block is saved on the L2 output memory. It is essential to bear in mind that compressed macro-block size is totally heterogeneous and therefore, it is not possible to enable a constant memory size in L2 output memory.

In other way, the image analysis must be an object of study since a camera is a sensor in a similar way as human eyes. Although, current computation is very rudimentary, it is the one that in the future must emulate how a brain works and it interprets information. In the future would be interesting to study how to compute several objects that are present in the images. We suggest tocompress several objects with different requirements. The learning and higher cognitive level sensations difficultly could be programmed in a series of scenes. Probably these scene extensions will not be reachable for any system for long; probably it would be too complex. When they are compressing in run-time create new optimal points different from already existing.

APPENDIX A: Methodology to obtain Processor Optimal Points

Nowadays, there is a large range of digital architectures (such as DSP, FPGAs, ASICs, general purpose processors or check chapter 1 of this thesis). Each one is intrinsically capable to achieve every task with different performance (execution time versus energy consumption). Historically, design engineers according to their experience recognized a valid HW/SW partitioning. Unfortunately, currently, any of these processors architectures can achieve all requirements (p.e; very low consumption, high performance, general purpose) at the same time. Moreover, this static view is not valid if we want to obtain different performances at different moments because we do not always work in the designed worst case working point. Although we obtain higher performance, this has an energy consumption penalty unacceptable for battery embedded systems. Therefore, the silicon industry develops different platforms with different heterogeneous processors connected to a segmented bus or, currently, it is carrying out a thorough research in the use of a NoC (Network on Chip) inside SoC.

Part of our work is based on optimize critical code parts to adequate them to the platform. This heterogeneity usually leads to complex and inefficient solutions for this kind of code implementations, though that these algorithms usually run in different platforms. Thus, in the chapter, we focus on just one complex task from the multimedia domain as is MPEG-4 MP compression. Later on, we will try to generalize obtained results to other types of processes since these algorithms have very differentiating code pieces, either more data-flow dominant or with higher control-flow behavior.

When one or more processors execute a set of concurrent tasks, a predefined scheduling algorithm must be applied to decide the task execution order. For a multiprocessor system, an assignment procedure must determine which processor will execute each task. We use the terminology defined in [210, 198].

Scenario Definition and High Level Energy Model

The compression of an image sequence is based on two types of compressions. Spatial Compression where is just compressed redundancies at image level, and the temporal compression where differences between consecutive images are looked for. Computation is done at macro-block level. A macro block is a rectangular block of 16 by 16 pixels of 8 bits (pixel). An I-image (I-VOB or Video OBject) type is just spatially compressed. These spatial components are transformed to frequential ones thanks to the Discrete Cosinus Transform (DCT) and resulting high frequencies are quantized since human

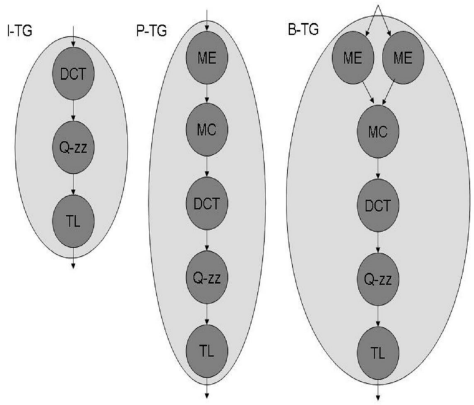


Figure 7.5: Task graphs for MPEG compression

vision is less sensitive to them. Afterwards, compressed information is zigzagged and Huffman coding is achieved to form the Transport Layer (TL).

In an image sequence, consecutive images usually have a lot of similarities (redundancies). Two algorithms are used to reduce these redundancies Motion Estimation (ME) that estimate a vector that minimizes image entropy and then these images are compensated (substrate) through the Motion Compensation (MC) algorithm. Finally, compensated images are also spatially compressed (DCT-basis).

There are two types of temporal compression images: P-type (P-VOB) that has just forward image compensation and B-type (B-VOB) that has two sorts of compensations backward and forward. The graph of different tasks (I, P or B tasks) with theirs subtasks (algorithms) that have to be accomplished to get a compressed image are shown in the graph in figure 7.5:

These sub-tasks graph represent pieces of code that contain mainly for loops with different data structures. These data structures are computed at macro block level after Code transformations (DTSE) are achieved mainly for minimizing the amount of transfers between external and internal memories, due to the fact that most energy is consumed during these data fetching. Another reason for working at macro block level is to have internal processor memory size and management independent of the image size that has to be compressed. Size of internal memory structures allows loading only few macro blocks. Data image pixels are independently loaded from external memories to internal ones and therefore, we can use any processor that can activate different number of FU of a processor to compute data in parallel.

These computing intensive loops are perfect to exploit loop buffering which is an effective technique to reduce energy consumption in the instruction memory hierarchy[256]. In any typical multimedia application, significant amount of execution time is spent in small program segments. Energy can be reduced by storing them in a small loop buffer instead of the big instruction cache (IL1). But, as more instruction level parallelism is extracted from the application, wider data-paths and wider loop buffers are

needed. Therefore, main bottleneck of wide VLIWs are data-path, register files and the interconnection network.

The energy model used in our flow takes profit of the fact that multimedia algorithms are based on some computationally intensive loops that are the kernel of the application. In the model, we consider the energy of the data computation and the transmission between the different memory layers as main sources of power consumption. In our case, the most computational intensive part of the algorithm is the encoder that requires in its inner loop a Multiply-Accumulate (MAC) operation. The SystemC description of the encoder allows the evaluation of different implementations with one, two, four and eight Functional Units (FU) hence being capable to exploit ILP and DLP. The energy and power figures were estimated from different technologies and scaled to 130nm technology (this is due because it is a work realized in 2006) with different values of VDD: 1V, 1.2V, 1,5V and 1,95V The technological parameters used for our case study come from reference [257].

Energy values Extracted from technology

From the technological information we can know the power consumption of the basic elements that form the hardware system, memories, MAC blocks, register etc. This low-level information is added to the high-level SystemC simulation to obtain different work points (execution time vs. power consumption). Therefore, derived from our SystemC model, Fig.7.7, shows the work points corresponding to different configurations. These points are obtained with different power and performance constraints. These points are obtained for different hardware architectures (number of FUs), supply voltages and running at different frequencies (150, 200, 450 and 600 Mhz).

For a given frequency, the most power hungry points in Fig.7.7 correspond to the large number of FUs with higher supply voltage but they need less time to compute any macro block. For example, to compress a macro block, we can use eight FUs for the MPEG kernel at high frequency and voltage, then processor is quite power hungry. This solution will nevertheless be the fastest one to compute the macro block. On the other hand, depending on the required periodic video frame rate and frame size, and the load of the actual data stream, we can switch between several Optimal Points that then become most advantageous in that given situation. In the actual implementation of the run-time selection these switches should then occur dynamically, then we can reduce voltage and frequency and then decrease power consumption using just one FU. Intermediate points were also obtained with diverse power-timing trade-offs.

Hardware Based platform OPW Results

To obtain optimal points in a HW platform, the first that it is necessary is to have information (energy, execution time) about the basic elements that form our model (how it is built): Memories, registers, elements MAC, adders, shifts, etc. This information is acquired at low level (target technology) and it is written in our SystemC platform model in high level to obtain Pareto points. These points were changing depending on

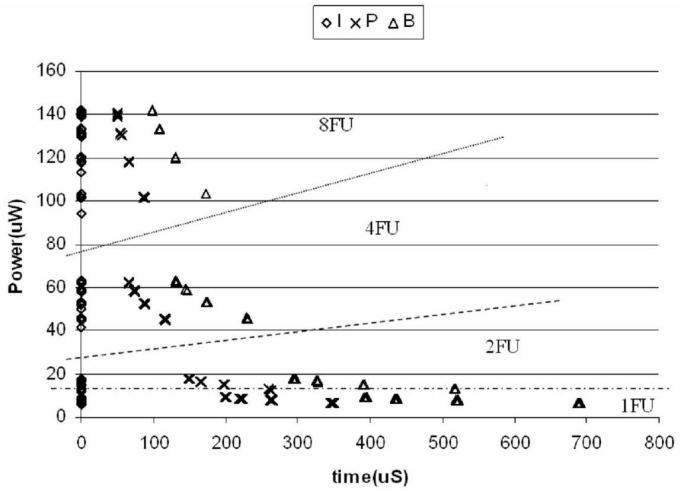


Figure 7.6: Optimal Work Points for different power-consumption vs. execution time trade offs using variable number of functional units obtained from SystemC modelling using Dynamic Voltage and Frequency Scaling.

the number of FUs(MACs blocks) that are in use and the type of macro-block to be compressed.

Taking as reference the high level results shown in Fig 7.7, we can analyze the real performance in Crisp-Trimaran target platform changing the number of functional units for a fixed voltage and frequency environment. 7.7 illustrates the outcome of this study. In the figure we can observe that specified architecture of the system with 12 functional units exist and theres is a. trade-of between time of execution and power necessary to execute it. Consequently that if we would like to accomplish the operations that we desire to complete, in our case the video compression of a macro block I, P or B. in the fastest way, the power consumption also will be the highest one. In the other way round, in the case we would like to compute the macro-blocks in low-power mode the time would increase and we just use a functional unit. Therefore, to execute the same application there is a range of work-points and not only a work point as it used to be when the engineers designed for the worst case scenario.

Instruction Set Processor Platform OPW Results

In a similar way, the pareto points are obtained for an ISP. In this case, there is a XML representation of the platform with technological values of this one (energy, execution time). Therefore, depending on the number of FUs that work in parallel and the type of image we can obtain a curve with the optimal power values.

In this section we presented a flow that goes from a C++ description of an application to a SystemC implementation. From the SystemC implementation we can derive some optimal work points in a similar way we can do it for an ISP. Also, the same application

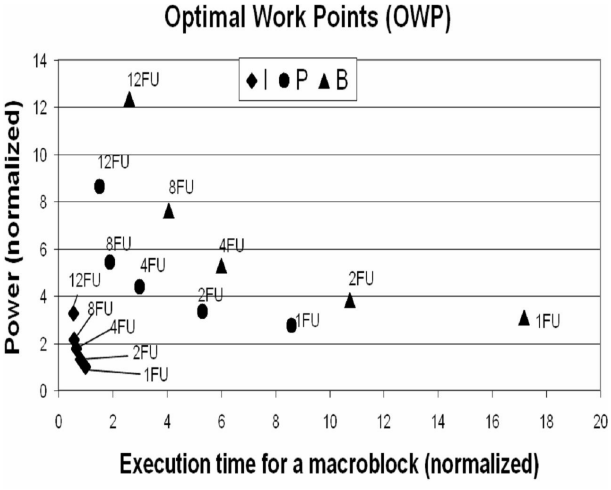


Figure 7.7: Optimal Work Points for different power-consumption vs. execution time trade offs using variable number of functional units obtained from the CRISP framework at 300 MHz and 1.8V.

has been mapped into five different platforms which show diverse hardware configurations and the time to accomplish the encoding of I, P and B macro blocks. New multimedia applications have recently increased in complexity and demand high performance at low power. Next generation of portable devices will present different energy-aware work points to optimize battery life. The previous results show that the OWP are not always easy to find. Different considerations as frequency and voltage (hence consumption) and the number of functional units can not be neglected if we want to maximize battery life. Run time dynamic voltage and frequency scaling will allow us to save lot of power. More effort has to be done for obtaining better quality-bit rates space for our application, so, we have to characterize MPEG models with more input standard images (at this moment we have just use foreman). From this work we published these two international conference papers [214, 215].

Future work will also involve the insertion on the flow of platform dependent transformations to optimize memory accesses and address generation. With these transformations IPC can increase considerably and the execution time will drop significantly. We also plan to extend our work to study a Network on Chip (NoC) [25] with several processors with different architectures and running at different voltages. The model of the NoC is already under development in SystemC. A first version is already realized and references are [190, 189]. This configuration will allow us to prove the usage of run time Dynamic Voltage Scaling and its power savings. In the following part of this chapter we explain the NoC and the relation with the MPEG-4 MP IP and some future work, that can be realized but it is not done yet due to the limited time to develop a thesis.

Bibliography

[1] J. Hennesy, The future of systems research, Computer Volume: 32, Issue: 8 (Aug. 1999) 27 –33.

[2] S. K. Moore, Multicore is bad news for supercomputers, IEEE Spectrum.

[3] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jureckska, L. Lavagno, C. Passarone, A. Sangiovanni-Vicentelli, E. Sentovich, K. Suzuki, B. Tabbara, Hardware-Software Co-Design of Embedded Systems: The POLIS Approach, Kluwer Academic Publisher, Boston /Dordrecht /London, 1997.

[4] A. Sangiovanni-Vicentelli, Defining platform based design, EEDesign.

[5] H.Chang, L.Cooke, G. M.Hunt, A.McNelly, L.Todd, Surviving the SOC revolution - A guide to platform-based design, Kluwer Academic Publishers, 1999.

[6] A. Ferrari, A. Sangiovanni-Vincentelli, System design: Traditional concepts and new paradigms, The Proceedings of the International Conference on Computer Design, ICCD ' Autin, TX, USA, pp 1-12, Oct. 1999 (Key-note address).

[7] e. a. Haibo Zeng, Design space exploration of automotive platforms in metropolis, Society of Automotive Engineers Congress.

[8] Digital Media System-on-Chip (DMSoC), TMS320DM357 Digital Media System-on-Chip (Web Site January 2009).
URL `www.ti.com SPRS553 NOVEMBER 2008, tms320dm357.pdf file`

[9] NXP Nexperia IO STB, Developmentkiy ST819, Advanced platform for building IP STB and dual functionality STB products, Nov. 06.

[10] Stratix ii dsp development board reference manual altera corporation, august 2006.

[11] Virtex-5 family overview, ds100 (v.4.3), june 18, 2008.

[12] G. Martin, e. a. H. Chang, Surviving in the SoC Revolution: A Guide to Platform Based Design, Kluwer Academic Publishers, Sept. 1999.

[13] A. Sangiovanni-Vicentelli, G. Martin, Platform-based design and software design methodology for embedded systems, IEEE Design & Test of Computers (2001) 23–33.

[14] Tensilica web site january 2009,.
URL `http://www.tensilica.com/`

[15] T. T. Ye, L. Benini, G. D. Micheli, Analysis of power consumption on switch fabrics in network routers, Proceedings of Design Automation Conference, 2002, DAC IEEE (June, 2002.) 524 –529.

[16] Y. Sheynin, E. Suvorova, F. Shutenko, Complexity and low power issues for on-chip interconnections in mpsoc system level design, Proceedings of the 2006 Emerging VLSI Technologies and Architectures (ISVLSI'06).

[17] S. Winegarden, Bus architecture of a system on a chip with user configurable system logic, IEEE Journal of Solid State Circuits 35, No. 3. (2000) 425–433.

[18] C. D. Thompson, A complexity theory for vlsi, ph.d. dissertation, dep. of computer science, pittsburg, pa . 178 p., Ph.D. thesis, Carnegie-Mellon Univ. (1980).

[19] H. Wang, L.-S. Peh, S. Malik, Power-driven design of router microarchitectures in on-chip networks, MICRO-36.

[20] J. Plosila, T. Seceleanu, P. Liljeberg, Implementation of a self-timed segmented bus, IEEE Design & Test of Computers.

[21] J. Y. C. et al., Segmented bus design for low-power systems, IEEE Trans. Very Large Scale Integration Systems vol. 7 - no 1 (Mar. 1999) pp.25–29.

[22] C. Kotsis, Interconnection Topologies for Parallel Processing Systems, PARS Mitteilungen, no 11. June 1993, pp.1-6.

[23] I. Ouveysi, a Wirth, On design of a survivable network architecture for dynamic routing: Optimal solution strategy and an efficient heuristics, European J, Operational Research 117, no. 1 (1999) 30–44.

[24] S. Hauck, Asynchronous design methodologies: An overview, Proc. IEEE, vol. 83 no. 1 (Jan. 1995) 69–93.

[25] L. Benini, G. D. Micheli, Networks on chip: a new soc paradigm, IEEE Computer, vol. 35, no. 1.

[26] L. Benini, G. D. Micheli, System-level power optimizations techniques and tools, ACM. Transactions on Design Automation for Embedded Systems (TODAES) 5, No. 2 (April 2000.) 115–192.

[27] C.-D. Chien, K.-P. Lu, Y.-M. Chen, J.-I. Guo, Y.-S. Chu, C.-L. Su, An area-efficient variable length decoder ip core design for mpeg-1/2/4 video coding applications, IEEE Transactions on Circuis and Systems for Video Technology vol.16, no 9.

[28] Amd to ship industry's first native x86 quad-core processors in august. amd.com. 2007-06-29. (january 2009).
URL $http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51\_104\_543\ 118193,00.html$

[29] Berkeley laboratories.
URL $http://parlab.eecs.berkeley.edu/publications$

[30] Amd product information (january 2009) (2009).
URL $http://www.amd.com/us-en/Processors/ProductInformation/$

[31] "fudzilla - westmere 32nm to improve nehalem features".
URL $http://www.fudzilla.com/index.php?option=com_content\&task=view\&id=9448\&Itemid=1$

[32] http://pc.watch.impress.co.jp/docs/2008/0326/kaigai02.pdf.

[33] Visio-intel cpu road map (febraury 2009).
URL $http://pc.watch.impress.co.jp/docs/2008/0321/kaigai_10.pdf$

[34] freescale "$http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=msc8156\&nodeid=0127950e5f5699$".

[35] http://focus.ti.com/lit/ml/sprb189b/sprb189b.pdf da vinci technical overview september 2008.

[36] M. Gschwind, D. Erb, S. Manning, M. Nutter), Cell processor an open source environment for cell broadband engine system software, IEEE Computer Vol. 40, No. 6.

[37] M. Gschwind, P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, T. Yamazaki, Synergistic processing in cell's multicore architecture, IEEE Micro Vol. 26, No. 2,.

[38] Beckton processor roadmap, january 09:.
URL `http://www.tgdaily.com/index2.php?option=com_content&do_pdf=1&id=38975`

[39] "fudzilla - westmere 32nm to improve nehalem features". fudzilla.com (september 16, 2008). retrieved on 2008-11-24.

[40] "ibm power 7 to be opteron socket compatible". the inquirer. retrieved on 2007-03-25.

[41] Deneb becomes phenom ii, amd promises overclocking for the masses, nordichardware, 18 november 2008.

[42] Intel dual-core processors (febraury 2009).
URL $http://www.intel.com/technology/computing/dual-core/index.htm$

[43] P. Microprocessors, .
URL $http://www.cpu-world.com/CPUs/K10/TYPE-Phenoml$

[44] M. Flynn, Some computer organizations and their effectiveness, IEEE Trans. Comput. , Vol. C-21, (1972.) pp. 948,.

[45] Report on penryn series improvements. (pdf). technology@intel magazine (october de 2006). consultado el 2007-08-28.

[46] G. de Haan, Progress in motion estimation for consumer video format conversion, IEEE Transactions on Consumer Electronics Vol. 46, No. 3.

[47] E. A. Lee, A programmable dsp overview, IEEE micro.

[48] "dsp selection guide 2007: Ti dsp products:making innovation possible", texas instuments, ssvv004.pdf.

[49] "embedded processors and dsp selection guide2007 edition", www.analog.com/processors.

[50] Buyer's guide to dsp processors, berkeley, california: Berkeley design technology, inc., 1994, 1995, 1997, 1999, 2001.

[51] F. Sijstermans, The trimedia processor: the price-performance challenge for media processing, IEEE International Conference on Multimedia and Expo.

[52] M. Sima, S. D. Cotofana, S. Vassiliadis, J. T. J. van Eijndhoven, K. A. Vissers, Pel reconstruction on fpga-augmented trimedia, IEEE Transactions in Very Large Scale Integration (VLSI) Systems, VOL. 12, NO. 6,.

[53] Mpc8220i system-on-a-chip based on powerpc core imaging applications specific standard product, rev 0, 5/30/2006,.

[54] R. L. Oliver Schliebusch, Heinrich Meyr, Optimized ASIP Synthesis from Architecture Description Language Models, Dordrecht: Springer. ISBN 978-1-4020-5685-7, 2007.

[55] R. L. Paolo Ienne (Ed.), Customizable Embedded Processors, San Mateo, CA: Morgan Kaufmann. ISBN 978-0-12-369526-0., 2006.

[56] K. K. Matthias Gries (Ed.), Building ASIPs: The Mescal Methodology, New York: Springer. ISBN 978-0-387-26057-0, 2005.

[57] P. O. de Beeck, F. Barat, M. Jayapala, R. Lauwereins, Crisp: A template for reconfigurable instruction set processors, In Proceedings of FPL.

[58] I. Group, Trimaran: An infraestructure for compiler research in instruction level parallelism, New York University (1998) www.trimaran.org.

[59] http://openmp.org/wp/ website april 09.

[60] S. P. Pacheco, Parallel Programming with MPI, Morgan Kauffmann Publishers, Inc., San Francisco, California, 1997, 418pp., ISBN 1-55860-339-5, 1997.

[61] J. Joven, O. Font-Bach, D. Castells-Rufas, R. Martinez, J. C. L. Teres, xenoc - an experimental network-on-chip environment for parallel distributed computing on noc-based mpsoc architectures, 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'08),.

[62] A, Arm926ej-s technical reference manual, release d, 26 january 2004, B, D, f (E 2004).

[63] Niosii processor reference handbook, altera corp. v.7.2 october 2007.

[64] Arm. , website january 09.
URL `www.arm.com`

[65] J. Tirado, M. Serra, A. Portero, J. Saiz, L. Ribas, J. Carrabina, Rapid prototyping platforms for reactive systems with polis-based hw-sw codesign approach, XV Design of Circuits and Integrated Systems Conference.

[66] F. Lisa, A. Portero, J. Carrabina, Designing sample applications for r.i.c. configurable platform, XV Design of Circuits and Integrated Systems Conference.

[67] F. Lisa, A. Portero, J. Carrabina, Prototyping systems for real time image processing algortihms, Design Automation and Test in Europe, DATE 2000, User Forum.

[68] B. Selic, The real-time uml standard: Definition and application, DATE.

[69] Matlabworks website: http://www.matlabworks.com.

[70] J. Ballagh, P. Athanas, E. Keller, Java Debug Hardware Models using JBits, 8th Reconfigurable Architectures Workshop, 2001.

[71] Y. Ha, B. Mei, P. Schaumont, S. Vernalde, R. Lauwereins, H. D. Man, Delopment of a design framework forplatform-independent networked reconfiguration of software and hardware, Lecture Notes In Computer Science archive Proceedings of the 11th International Conference on Field-Programmable Logic and Applications 2147 (2001, ISBN:3-540-42499-7) 264 – 274.

[72] J. E. H. III, A Device-Level FPGA Simulator, Master of Science in Computer Engineering, University of Virginia, June 2004.

[73] A. Poetter, J. Hunter, C. Patterson, P. Athanas, B. Nelson, N. Steiner, Jhdlbits: The merging of two worlds, FPL, LNCS 3203 (2004) 414–423.

[74] S. Singh, P. James-Roxby, Lava and jbits: From hdl to bitstream in seconds, Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCC'01) 0-7695-2667-5.

[75] K. B., M. S. Kent, Context switching in a hardware/software co-design of the java virtual machine, DATE.

[76] J. C. D. Castells-Rufas, J. Joven, A validation and performance evaluation tool for protonoc, International Symposium on System-on-Chip 2006 (SOC2006). Tampere, Finland.

[77] java.lang.system android documentation. google.
URL $http://code.google.com/android/reference/java/lang/System.html.Retrievedon2008-12-13$

[78] Native c++ game s-tris2 running on android. android-developers mailing list. 2007-11-16. (visited january 2009).
URL $http://groups.google.com/group/android-developers/browse_thread/thread/f31003bbed8bf7a9/$

[79] S.Liao, S. Tjiang, R.Gupta, An efficient implementeation of reactivity modeling hardware in the scenic design environment, 34th Design Automation Conference, Anaheim (1997) 70–75.

[80] R. Roth, D. Ramanathan, A high level hardware design methodology using c++, 4th High Level Design Validation and Test workshop, San Diego (1999.) 73–80.

[81] Open systemc iniciative website : http://www.systemc.org.

[82] http://www.cynapps.com/ (forteds cynthesizer).

[83] D. Gajski, D. Jianwen Zhu, A. R., Gerstlauer, S. Zhao, SpecC: Specification Language and Methodology, 2000.

[84] A. Gerstlauer, D. R., J. Peng, D. Gajski, System Design A Practical Guide with SpecC, Kluwer Academics, Hardcover ISBN: 978-0-7923-7387-2, 2001.

[85] website [www.cynapps.com] march 2006, currently (jan 09) www.forteds.com.

[86] website visited january 2009.
URL `SystemVerilog www.systemVerilog.org`

[87] lssam Damaj, J. Hawkins, A. Abdallah, Mapping high level algorithms onto massively parallel reconfigurable hardware, in: Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference on, 2003.

[88] Draft Standard SystemC AMS Extensions Language Reference Manual, Dec03 2008 website: (visited January 09).
URL `http://www.systemc-ams.org`

[89] Digital video quality, stefan winkler, wiley, march 2005, isbn 0-470-02404-6.

[90] Itu-t rec. j.247 (08/08) objective perceptual multimedia video quality measurement in the presence of a full reference.

[91] Itu-t rec. j.246 (08/08) perceptual audiovisual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference.

[92] M. Tools, Applications, Quantified PQoS assessment based on fast estimation of the spatial and temporal activity level, Springer Netherlands, 2007.
URL `http://www.springerlink.com/content/f8v2p4r852266415`

[93] Reference: Iso/iec jtc1/sc29/wg11 (june 1996).

[94] 13818: 'generic coding of moving pictures and associated audio (mpeg-2) itu-t rec. h.262, iso/iec 13818-2, " generic coding of moving pictures and associated audio ", draft int. standard, oct. 1994.

[95] W. Pennebaker, J. Mitchell, Jpeg still image data compression standard, Van Nostrand Reinhold.

[96] www.mpeg.org the moving picture experts group (mpeg) web site.

[97] V. Bhaskaran, K. Konstantinides, Image and Video Compression Standards - Algorithms and Architectures, Kluwer Academic Publishers (Boston), 1996.

[98] P. Farrelle, Recursive Block Coding for Image Data Compression, Springer-Verlag, 1990.

[99] P. Symes, Video Compression Demystified, McGraw-Hill, December 2000.

[100] F. Pereira, T. Ebrahimi, The MPEG-4 book, IMSC Press Multimedia Series.

[101] R. Koenen, Mpeg-4 - multimedia for our time, IEEE Spectrum Vol. 36, No. 2 (February 1999) 26–33.

[102] Iso / iec 14496, amendment 1, information technology - coding of audio- visual objects - part 2: Visual 2001.

[103] B. Manjunath, P. Salembier, T. Sikora (Eds.), Introduction to MPEG-7: Multimedia Content Description Interface, Wiley & Sons ISBN 0-471-48678-7, April 2002.

[104] H. Kosch, Distributed Multimedia Database Technologies, CRC Press - ISBN 0-8493-1854-8, January 2004.

[105] G. S. (Editor), S. K. (Editor): (Eds.), Multimedia Content and the Semantic Web: Standards, Methods and Tools, Wiley & Sons - ISBN 0-470-85753-6, May 2005.

[106] H.-G. Kim, N. Moreau, T. Sikora, MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval, Wiley & Sons - ISBN 0-470-09334-X, October 2005.

[107] http://mpeg.chiariglione.org website april 2009.

[108] K. Diepold, F. Pereira, W. Chang, Mpeg-a: Multimedia application formats, IEEE Multimedia October.

[109] I. E. G. Richardson, Video Codec Design: Developing Image and Video Compression Systems, John Willey & Sons Ltd., 2002.

[110] K. Rao, R. Yip, Discrete Cosine Transform-Algorithms, Advatages, Applications, Academic Press, 1990.

[111] N. Ahmed, T.Natarajan, K.Rao, Discrete cosine transform, IEEE Transactions on Computing Vol. 23 (January 1974) 90–93.

[112] W. Chen, C. Smith, S. Fralick, A fast computational algorithm for the discrete cosine transform, IEEE Transactions on Communications,, . Vol. 38.

[113] S. LLoyd, Least squares quantization in pcm, IEEE Transactions on Information Theory Vol. 28, No. 2 (March 1982) 129–137.

[114] J.Max, Quantizing for mimimum distortion, Transactions of IRE vol. 6 (March 1960) .7–12.

[115] B. Chitpraset, K. Rao, Human visual weighed progressive image transmission, IEEE Transactions on Communications Vol. 39.

[116] I. Witten, R. Neal, J.Clearly, Arithmetic coding for data compression, Communications of the ACM 30(6).

[117] A. Lambrechts, P. Raghavan, M. Jayapala, F. Catthoor, D. Verkest, Interconnect-exploration for energy vs performance tradeoffs for coarse grained reconfigurable architectures, IEEE Transactions on VLSI.

[118] P. Raghavan, A. Lambrechts, J. Absar, M. Jayapala, F. Catthoor., Coffee: Compiler framework for energy-aware expoloration., In Proc of HiPEAC.

[119] S. K., Proceedings of the Tenth Systems Administration Conference (LISA 96), Chicago, IL, (October 1996) 161–170.

[120] D. C. Pham, et al, Overview of the architecture, circuit design, and physical implementation of first-generation cell processor, IEEE Journal of SSC.

[121] Texas instruments, "tms320c6000 cpu and instruction set reference guide", spru189f, october 2000.

[122] D. Baumgartner, P. Rossler, W. Kubinger, Performance benchmark of dsp and fpga implementations of low-level vision algorithms, Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on,.

[123] ISO, International standard iso/iec 14496-2 2nd edition 2001-12-01 it (coding of audio-visual objects) part 2, Tech. rep., ISO (2001).

[124] P. Ranganathan, S. Adve, N. Jouppi, Performance of image and video processing with general-purpose processors and media isa extensions, in: Proceedings of the 26th International Symposium on Computer Architecture, 2-4 May 1999, pp. 124–135.

[125] J. Turley, Embedded processors by the numbers, in: Embedded Systems Programming, 12(5), 1999.

[126] V. Lappalainen, T. Himillinen, P. Liuha, Overview of research effort on media isa extensions and their usage in video coding, IEEE Trans. Circ. Syst. Video Technology vol. 12, no. 8, (2002) 660–670.

[127] P. Tseng, Y. Chang, Y. Huang, H. Fang, C. Huang, L. Chen, Advances in hardware architectures for image and video coding - a survey, Proceedings of the IEEE Volume 93, Issue 1, (Jan. 2005) 184–197.

[128] J. M. Rabaey, M. J. Ammer, J. L. da Silva Jr., D. Patel, S. Roundy., Hoc ultra-low power wireless networking, Computer 33,ï¿œ Issue 7, (July 2000) 42 – 48.

[129] M. Wan, J. Rabaey, et. al, Design methodology of a low-energy reconfigurable single-chip dsp system, Journal of VLSI Signal Processing.

[130] M. Seltzer, D. Krinsky, K. Smith, X. Zhang, The case for application-specific benchmarking, in: Proceedings of the Seventh Workshop on Hot Topics in Operating Systems, 1999.

[131] P. Pirsch, M. Berekovic, H.-J. Stolberg, J. Jachalsky, Vlsi architectures for mpeg, International Symposium on VLSI Technology, Systems, and Applications.

[132] J. C. Chen, C.-F. Shen, , S.-Y. Chien, Coarse-grained reconfigurable image stream processor for digital still cameras and camcorders, in: IEEE Custom Integrated Circuits Conference (CICC), 2007.

[133] M. Wan, H. Zhang, V. George, M. Benes, A. Abnous, V. Prabhu, J. Rabaey, Design methodology of a low-energy reconfigurable single-chip dsp system, Journal of VLSI Signal Processing.

[134] A. Portero, G. Talavera, J. Carrabina, F. Catthoor, Data-dominant application implementation in multiplatform for energy-flexibility space exploration, Transactions in Very Large Scale Integration.

[135] P. Kuhn, Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwer A.P.

[136] F. Pereira, T. Ebrahimi, The MPEG-4 book, IMSC Press Multimedia Series.

[137] A. Lambrecths, T. V. Aa, M. Jayapala, A. Leroy, G. Talavera, A. Shickova, F. Barat, D. V. Francky Catthoor, G. Deconinck, H. Coporaal, F. Robert, J. C. Bordoll., Design style case study for compute nodes of a heterogenenous noc platform., In 25th IEEE Real-Time Systems Symposium (RTSS).

[138] J. R. Jain, A. K. Jain, Displacement measurement and its application in interframe image coding, IEEE Transactions on Communications Vol. Com-29, NO.

[139] V. Bhaskaran, K. Konstantinides, Image and video compression standards : Algorithms and architectures, Kluwer, (June 1997,) 454.

[140] Micron, 1-2.5 inch 5mp cmos digital image sensor mt9p031, Tech. rep., www.micron.com (2007).

[141] F. C. et al, Data access and storage management for embedded programmable processors, Kluwer AP., 2002.

[142] D. L. Harris, S. F. Oberman, M. A. Horowith, Srt division architectures and implementations, in: Proceedings of the 13th Symposium on Computer Arithmetic (ARITH '97), ISBN:0-8186-7846-1 IEEE Computer Society Washington, DC, USA, 1997.

[143] Document, cynthesizer users guide for cynthesizer version 5.4 from forteds www.forteds.com.

[144] Faraday UMC technology libraries http://freelibrary.faraday-tech.com/.

[145] A. Portero, G. Talavera, M. Moreno, B. Martínez, J. Saiz, M. Montón, J. Carrabina, Multiplatform video encoder implementation for energy-flexibility space exploration, Microprocessors and Microsystems Embedded Hardware Ddesign MICPRO (to appear) elsevier.

[146] Powerplay early power estimator user guide for stratix ii, stratix ii gx, & hardcopy ii document version: 1.2.

[147] M. M. M. o. o. o, M. H. B, J. Carrabina Bordoll, SÃntesis de canales tlm para procesador nios-ii, JCRA (Spanish).

[148] Command reference for buildgates synthesis and cadence pks timing analysis product version 5.15 january 2005, Tech. rep., Cadence (January 2007).

[149] A. P. ChipAttack.com, Chip attack http://chipattack.com (document will be upload if accepted) (2009).

[150] S. S. Muchnick, Advanced compiler design and implementation, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[151] K. Kennedy, J. R. Allen, Optimizing compilers for modern architectures: a dependence-based approach., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[152] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, Compilers: Principles, Techniques, and Tools (2nd Edition), Addison Wesley, Boston, MA, USA, 2006.

[153] F. Catthoor, et al., Custom Memory Management Methodology, Kluwer AP, 1998.

[154] F. Catthoor, E. al, Data Access and Storage Management for Embedded Programable Processors, 2002.

[155] S. Wuytack, F. Catthoor, L. Nachtergaele, H. D. Man, Power exploration for data dominated video applications, in: ISLPED '96: Proceedings of the 1996 international symposium on Low power electronics and design, IEEE Press, Piscataway, NJ, USA, 1996, pp. 359–364.

[156] H. Falk, P. Marwedel., Source Code Optimization Techniques for Data Flow Dominated Embedded Software., Springer, 2004.

[157] H. Falk, Control flow driven code hoisting at the source code level, In ODES'05: Proceedings of The 3rd Workshop on Optimizations for DSP and Embedded Systems.

[158] H. Falk, P. Marwedel., Control flow driven splitting of loop nests at the source code level., In DATE '03: Proceedings of the conference on Design, Automation and Test in Europe, pages , Washington, DC, USA. IEEE Computer Society. (2003) 410–415.

[159] White paper digital media system-on-chip (dmsoc) tms320dm6441 sprs359d 2005-revised march 2008.

[160] Texas instruments, "tms320c6000 programmer's guide", 2000.

[161] Texas instruments, "code composer studio v3.1 ide getting started guide", http://www-s.ti.com/sc/psheets/spru509g/spru509g.pdf.

[162] T. D. M. S. on Chip SPRS283F, Tech. rep., Texas Instruments www.ti.com (December 2005 revised March 2008).

[163] M. Bhatnagar, Tms320dm6441 power consumption summary - spraau3, Tech. rep., Application Report, Texas Instruments, (April 2008).

[164] M. Berekovic, A. Kanstein, B. Mei, Mapping mpeg video decoders on the adres reconfigurable array processor for next generation multi-mode mobile terminals, in: GSPX, 2006.

[165] S. Agarwala, et al, A 600-mhz vliw dsp, IEEE Journal of Solid-State Circuits Vol. 37, No. 11,.

[166] G. Talavera, M. Jayapala, J. Carrabina, F. Catthoor, Address generation optimization for embedded high-performance processors: A survey, Journal of Signal Processing Systems for Signal Image and Video Technology 53 (2008) 271–284. doi:10.1007/s11265-008-0165-y. URL `http://www.springerlink.com/content/7h02101018340512/`

[167] R. Banakar, S. Steinke, B.-S. Lee, M. Balakrishnan, P. Marwedel, Comparison of cache- and scratch-pad-based memory systems with respect to performance, area and energy consumption, Technical Report 762, University of Dortmund, http://citeseer.ist.psu.edu/banakar01comparison.html.

[168] J. Absar, F. Catthoor, Analysis of scratch pad and cache performance using statistical analysis, Asia and South Pacific Design Automation Conference (ASP-DAC).

[169] T. V. Aa, M. Jayapala, F. Barat, G. Deconinck, F. Catthoor, H. Corporaal, Instruction buffering exploration for low energy vliws with instruction clusters, in Proc. IEEE Asia and South Pacific Design Autom. Conf. (ASPDAC), Yokohama, Japan, (Jan. 2004.) pp.825–830,.

[170] T. VanderAa, M. Jayapala, H. Corporaal, F. Catthoor, G.Deconinck, Instruction transfer and storage exploration for low energy vliws, IEEE Workshop on Signal Processing Systems Design and Implementation SIPS '06. I (Oct.2006) 292–297.

[171] L. H. Lee, W. Moyer, J. Arends, Instruction fetch energy reduction using loop caches for embedded applications with small tight loops, in Proc of ISLPED.

[172] M. Jayapala, F. Barat, T. V. Aa, F. Catthoor, H. Corporaal, G. Deconinck., Clustered loop buffer organization for low energy vliw embedded processors, IEEE Trans. on Computers (June 2005) 54(6):672–683,.

[173] J. A. Fisher, P. Faraboschi, , C. Young., Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools, Morgan Kaufmann, 2004.

[174] G. Talavera, M. Jayapala, J. Carrabina, F. Catthoor, Address generation optimization for embedded high-performance processors: A survey, Journal of Signal Processing Systems for Signal Image and Video Technology.

[175] P. Raghavan, A. Lambrechts, M.Jayapala, F.Catthoor, D.Verkest, H. Corporaal, Very wide register: An asymmetric register file organization for low power embedded processors, In "DATE '07: Proceedings of the conference on Design.

[176] P. Raghavan, A. Lambrechts, M. Jayapala, F. Catthoor, D. Verkest, Distributed loop controller for multi-threading in uni-threaded ilp architectures, (to appear) IEEE Transactions on Computers.

[177] ALTERA, Hardcopy series Handbook, 101 innovation serie San Jose California, CA95134, 2008.

[178] S. Segui, L. Igual, F. Vilarino, P. Radeva, C. Malagelada, F. Azpiroz, J. Vitri, Disfunctions using video capsule endoscopy, A. Gasteratos, M. Vincze, and J.K. Tsotsos (Eds.) ICVS 2008, LNCS 5008, Springer-Verlag Berlin Heidelberg 2008 (2008) 251 260.

[179] A. Wolfe, Hdtv - ready for the long drive?, Spectrum, IEEE Volume 40, Issue 6 (June 2003) 13– 15 Digital Object Identifier 10.1109/MSPEC.2003.1203077.

[180] "zpower to demonstrate the first silver-zinc battery for mobile electronics at the intel developer forum 2008". http://www.zpowerbattery.com/pdf/idfon 2008-08-24 web page february 2009.

[181] Micron and aptina imaging datasheets, http://www.slac.stanford.edu/econf/c020909/efslide.pdf slide 14 (web site february 2009).

[182] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, Y. F. H. a, Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards, Computer Communications, elsevier.

[183] A. J. Fischer, A. A. Allerman, M. H. Crawford, K. H. A. Bogart, S. R. Lee, R. J. Kaplar, W. W. Chow, S. R. Kurtz, K. W. Fullmer, J. J. Figiel, Room-temperature direct current operation of 290 nm light-emitting diodes with milliwatt power levels, Appl. Phys. Lett. 84, 3394 (2004); DOI:10.1063/1.1728307.

[184] J. A. Fisher, P. Faraboschi, C. Young, Embedded computing, Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools., Morgan Kaufmann, 2004.

[185] C. A. Zeferino, A. A. Susin, Socin: A parametric and scalable network-on-chip, SBCCI'2003, IEEE CS Press (2003) pp.169–174.

[186] C. A. Zeferino, M. E. Kreutz, A. A. Susin, Rasoc: A router soft-core for networks-on-chip, DATE'2004 - Designer's Forum, IEEE CS Press.

[187] M. Monton, B. Martinez, J. Carrabina, Sintesis de canales tlm para procesador nios-ii", VIII Jornadas de Computación Reconfigurable y Aplicaciones, (JCRA-2008).

[188] A. Bobik, Handbook of Image and Video ProcessingCompression Standards, Algorithms and Architectures., Academic Press, 2000.

[189] A. Portero, R. Pla, A. Rodriguez, J. Carrabina, Noc design of a video encoger in a multiprocessor system on chip solution, The 17th International Conference on ICM, December 13-15 (2005) 198 203.

[190] A. Portero, R. Pla, J. Carrabina, Systemc implementation of a noc, Industrial Technology ICIT 2005. IEEE International Conference on 14-17 December (2005) 1132 – 1135.

[191] M. Geilen, T. Basten, B. Theelen, R. Otten, An algebra of pareto points, Proceedings of the Fifth International Conference on Application of Concurrency to System Design (ACSD'05) 1550-4808/05.

[192] V. Pareto, Manuale di Economia Politica, Translated into English by A.S. Schwier (1971), Manual of Political Economy, MacMillan, London., 1971(1906).

[193] M. Yukish., Algorithms to identify pareto points in multi-dimensional data sets. phd thesis, Ph.D. thesis, Pennsylvania State University (August 2004.).

[194] C. Sun, X. Qi, O. Li, Pareto-mec for multi-objective optimization, 0-7803-7952-7/03 IEEE. (2003) 321–328.

[195] M. Gries, Methods for evaluating and covering the design space during early design development. integration, the VLSI Journal 38(2) (2004.) 131–183.

[196] F. Thoen, F. Catthoor, Modeling, Verification and Exploration of Task-Level Concurrency in Real-Time Embedded Systems, Kluwer, 2000.

[197] R. Szymanek, F. Catthoor, K. Kuchcinski, Time-energy design space exploration for multilayer memory architectures, In Proc. of DATE . IEEE (2004) 318–323.

[198] P. Yang, F. Catthoor, Pareto-optimization-based runtime task scheduling for embedded systems, In Proc. Of CODES+ISSS (2003) 120–125.

[199] T. Givargis, F. Vahid, J. Henkel, System-level exploration for pareto-optimal configurations in parameterized system-on-a-chip, IEEE Trans. VLSI Syst. 10(4) (August 2002) 416–422.

[200] M. Ehrgott, X. Gandibleux, An annotated bibliography of multi-objective combinatorial optimization. technical report 62/2000, fachbereich mathematik, Tech. rep., Universitat Kaiserslautern, Kaiserslautern, Germany (2000.).

[201] C. Mattson, A. Mullur, A. Messac, Minimal representation of multiobjective design space using a smart pareto filter, In Proc. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.

[202] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. G. D. Fonseca., Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation 7(2) (April 2003) 117–132.

[203] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, New York, 2001.

[204] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Trans. on Evolutionary Computation 3(4) (November 1999.) 257–271.

[205] A. Baykasoglu, S. Owen, N. Gindy, A taboo search based approach to find the pareto optimal set in multiple objective optimisation, Journal of Engin. Optimization 31 (1999.) 731–748.

[206] J. Bentley, Multidimensional command-and-conquer, Communications of the ACM 23(4): (April 1980.) 214–229.

[207] R. Finkel, J. Bentley, Quad trees: A data structure for retrieval on composite keys, Acta Inf. 4:1-9.

[208] F. Glover, Tabu search: A tutorial. interfaces, 20(4) (1990) 74–94,.

[209] M. Sun, R. E. Steuer., Quad-trees and linear lists for identifying nondominated criterion vectors, ORSA Journal on Computing 8(4) (1996.) 367–375,.

[210] S. V. et al., System-scenario-based design of dynamic embedded systems, ACM Transactions on Design Automation of Electronic Systems Vol. 14, No. 1, Article 3, Pub.

[211] T. Instruments, Omap3, family of multimedia applications processors (2007).

[212] J. L. Hennessy, D. A. Patterson, Computer Architecture: A Quantitative Approach. 3rd edition, Morgan Kaufmann, 2002.

[213] A. A. et al, Profile-based dynamic voltage scheduling using program checkpoints, in Proc. IEEE Design, Automation and Test in Europe Conference.

[214] A. Portero, G. Talavera, M. Montón, B. Martinez, F. Catthoor, J. Carrabina, Dynamic voltage scaling for power efficient mpeg-4sp implementation, IEEE-ASAP 06 Appliocation-Specific Systems, Architectures and Processors. Steamboat Spring , Colorado (Set 11-13. 2006) 257–260.

[215] A. Portero, G. Talavera, M. Montón, M. M. F. Catthoor, J. Carrabina, Energy-aware mpeg-4 single profile in hw-sw multi-platform implementation, IEEE- SOCC06 System On Chip Conference, Austin Texas, Set 24-27 (2006) 13–16.

[216] J. Saiz, M.Serra, J.Tirado, A.Portero, J. Carrabina, Un camino desde el codiseño hw/sw hasta las plataformas de prototipado para computación reconfigurable, Jornadas sobre Computación Reconfigurable y Aplicaciones.

[217] A. Portero, F. Lisa, J. Carrabina, Automatic interface synthesis for ip-based design, XV Design of Circuits and Integrated Systems Conference.

[218] A. Portero, J. Carrabina, Compromiso entre comunicación y computación en sistemas hw-sw reconfigurables, Jornadas sobre Computación Reconfigurable y Aplicaciones.

[219] J. Carrabina, L. Ribas, J.Saiz, A.Portero, M.Serra, J.Ordeix, P.Marti, Descripción de la computación y su implicación reconfigurable, Jornadas sobre Computación Reconfigurable y Aplicaciones Año: 2002 ISBN: 84-699-9448-4.

[220] J. Carrabina, L. Ribas, A. Portero, B. Martinez, M. Monton, L. Teres, R. Martinez, D. Castells, R. Puig, M. Serra, Plataformes de prototipat rápid revista, I Jornades de Codisseny Hardware-Software, Universidad de Vic.

[221] O. Ferraz, M. Monton, R. Juvanteny, A. Portero, J. Carrabina, Flexicamera intelligent and flexible camera, Design of Circuits and Integrated Systems Conference.

[222] A. Portero, R. Pasko, G. Vanmerbeek, S. V. Diederik Verkest, J. Carrabina, Flujo de codiseño para la implementación de algoritmos de compresión de imágenes para sistemas hw-sw reconfigurables, Jornadas sobre Computación Reconfigurable y Aplicaciones, ISBN: 84-699-9448-4.

[223] A. Portero, J. Carrabina, Un exemple de comunicació vs computació dŽalgoritmes de compressió dímatges per sistemes en un xip, I Jornades de Codisseny Hardware-Software, Universidad de Vic.

[224] A. Portero, P. Marchal, J. I. Gomez, L. Piñuel, F. Catthoor, J. Carrabina, A study of trade offs in inter-frame compression mpeg-4 for a multiprocessor platform, XIX Conference on Design of Circuits and Integrated, France, November 24-26.

[225] A. Portero, J. V. Moyano, R. Pla, O. Navas, J. Carrabina, Implementación de algoritmos mpeg usando systemc, FPGAs Computación y Aplicaciones JCRA IV Jornadas de Computación Reconfigurable y Aplicaciones, Bellaterra 13-15 Setiembre de 2004-09-23.

[226] L. Ribes, A. Portero, Especificación de concurrencia y partición temporal (pipelining) en systemc, FPGAs Computación y Aplicaciones JCRA IV Jornadas de Computación Reconfigurable y Aplicaciones Año: Bellaterra 13-15 Septiembre de 2004-09-23.

[227] D. Castells, M. Monton, R. Pla, D. Novo, A. Portero, O. Navas, Comparing design flows for structural system level specifications facing fpga platforms, XIX Conference on Design of Circuits Integrated Systems, France, November 24-26.

[228] A. Portero, O. Navas, J. Carrabina, Study of high level design methodologies for a mpeg frames i compressor directed to fpga implementation, ICIT 2004 IEEE International Conference on Industrial Technology, Año: Dec 8-10 Hammamet Tunis.

[229] A. Portero, O. Navas, J. V. Moyano, M. Bonamusa, J. Escrig, J. Carrabina, A mpeg 2 frame i encoger developed with systemc vs. matlab dsp builder, GSPx, The Internacional Embedded Solution Event September 27-30 Santa Clara Convention Center, Santa Clara, CA. EEUU.

[230] A. Portero, O. Navas, J. Carrabina, Hw-sw design methodologies used for a mpeg video compressor synthesis, ICM 2004, IEEE 16 th International Conference on Microelectronics 06- 08 December , Sfax Tunis.

[231] A. Portero, L. Ribas, J. Carrabina, Hardware synthesis of parallel machines from systemc, FDL Forum on Specfication & Design Language, Laussane, Switzerland.

[232] A. Portero, G. Talavera, F.Catthoor, J. Carrabina, A study of a mpeg-4 codec in a multiprocessor platform, IEEE-ISIE 06 IEEE- International Symposium on Industrial Electronics, Montreal, Canada, 9-13 July.

[233] R. Aragonés, J. Oliver, B. Lorente, A. Portero, J. Saiz, C. Ferrer, A generic signal processor for frequency sensor data acquisition, DCIS XXIIConference on Design of Circuits and Integrated Systems.

[234] M. Montón, A. Portero, M. Moreno, B. Martínez, J. Carrabina, Mixed sw/systemc soc emulation framework, IEEE International Symposium on Industrial Electronics. (ISIE) . Vigo, 4-7 June 2007. ISBN: 1-4244-0755-9.

[235] antoni portero, guillermo talavera, G. Talavera, M. Moreno, B. Martinez, J. Saiz, M. Monton, J. Carrabina, F. Catthoor, Multiplatform video encoder implementation for energy-flexibility space exploration, MICPRO, elsevier (under revision).

[236] A. Portero, G. Talavera, J. Carrabina, F. Catthoor, Data-dominant application implementation in multiplatform for energy-flexibility space exploration, IEEE Transactions on Very Large Scale Integration (under revision).

[237] J. Saiz, A. Portero, R. Aragonés, J. Aguiló, M. Rullán, Experiencia de adaptación al eees de sistemas digitales en ingeniería informática de la uab, I Jornadas Nacionales de Intercambio de Experiencias Piloto de Implantación de Metodologías ECTS.

[238] A. Portero, J. Saiz, R. Aragonés, M. Rullan, J. Aguiló, E. Valderrama, Convergencia hacia el eees en sistemas digitales de ingeniería informática, CIDUI 2006.

[239] A. Portero, J. Saiz, R. Aragonés, M. Rullán, E. Valderrama, J. Aguiló, Adopting new competences experience in the face of engineering learning, IEEE First International Conference on E-Learning in Industrial Electronics ICELIE 2006. Hammamet, Tunisia. 18-20.

[240] A. Portero, D. Dent, Developing system-on-chip experioence in a higher education environment, ISNN: 0963-3308 IEE Electronics & Communications.

[241] R. Aragonés, J. Saiz, A. Portero, M. Rullán, J. Aguiló, Experiencia de innovación docente siguiendo las directrices del eees en la enseñanza del diseño digital., Revista Latinoamericana de Tecnología Educativa (RELATEC) ISSN: 1695-288X. vol. 5, nº 2 (2006) 203–222.

[242] J. Saiz, A. Portero, R. Aragonés, M. Rullán, J. Aguiló, Máquinas algorítmicas: una metodología para su aprendizaje práctico a través de littleproc, IEEE-Revista Iberoamericana de Tecnologías del Aprendizaje (IEEE-RITA).

[243] A. Portero, J. S. Alcaine, G. Talavera, R. Aragonés, M. Rullán, J. Aguiló, E. V. ., Aplicación del plan piloto en sistemas digitales en ingeniería informática siguiendo las directivas del eees, Libro de resúmenes del TAEE 2006, ISBN: 84-689-9590-8 (2006) pag. 87–88.

[244] A. Portero, J. Saiz, R. Aragonés, M. Rullan, J. Aguiló, E. Valderrama, Transforming spanish student attitude in the face of engineering learning, Proceedings of the WCCEE 2006. Viena.

[245] X.Ji, Energy-efficient video transmission over multi-user wireless networks, em Doctoral dissertation, ESAT/EE Dept., K.U.Leuven, Belgium, Sep. 2008.

[246] X. Ji, S. Pollin, G. Lenoir, G. Lafruit, A. Dejonghe, F. Catthoor, Multi-user motion jpeg200 over wireless lan: run-time performance-energy optimization with application-aware cross-layer scheduling, international packet video, Workshop.

[247] e. a. X. Ji, Energy efficient bandwidth allocation for multi-user video streaming over wlan, in ICASSP.

[248] X. Ji, G. Lafruit, I. Moccagatta, S. Pollin, A. Dejonghe, F. Catthoor, Network-adaptive and energy-efficient multi-user video communication over qos enabled wlan,, 15th European Signal Processing Conference (EUSIPCO), Pozna, Poland,.

[249] G. Al-Kadi, A. S. Terechko, A hardware task scheduler for embedded video processing, HiPEAC and Lecture Notes in Computer Science, Springer Berlin / Heidelberg.

[250] E. Salminen, T. Kangas, T. D. Hämäläinen, The impact of communications on the scalability of the data-parallel video encoder on mpsoc, System-on-Chip, IEEE International Symposium on.

[251] D. Culler, J. Singh, A. Gupta, Parallel computer architecture- a hardware/software approach, Morgan Kaufmann.

[252] A. Andriahanteniana, H. Charlery, A. Greiner, L. Mortiez, C. A. Zeferino, Spin: a scalable packet switched, on-chip micro-network, in DATE Munich, March 2003 70–77.

[253] P. Pande, C.Grecu, A. I. M. Jones, R. Saleh, Performance evaluation and design tradeoffs for network on chip interconnect architectures, IEEE Transactions on Computers 54 (August 2005) 1025–1040.

[254] A. Leroy, D. Milojevic, F. Diederik Verkest, M. Frederic Robert, F. Catthoor, Concepts and implementation of spatial division multiplexing for guaranteed throughput in networks-on-chip, IEEE TRANSACTIONS ON COMPUTERS VOL. 57, NO. 9,.

[255] J. (2008-01-01)., 204-pin ddr3 sdram so-dimm specification.

[256] S. Lee, S. I. Chae, Motion estimation algorithm using low resolution quantization, IEE Electronic Letters , vol 32, no 7, 28th.

[257] I. Group, Trimaran: An infraestructure for compiler research in instruction level parallelism, New York University.