Universidad Miguel Hernández de Elche

Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica

Design and implementation of FPGA-based video encoding accelerators

PhD Thesis

A dissertation for the degree of Doctor in Industrial and Telecommunication Technologies by:

Estefanía Fátima Alcocer Espinosa

Advisors:

Otoniel Mario López Granado

Roberto Gutiérrez Mazón

Universidad Miguel Hernández de Elche

Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica

Design and implementation of FPGA-based video encoding accelerators

Tesis Doctoral

Memoria presentada para optar al grado de Doctora en Tecnologías Industriales y de Telecomunicación por:
Estefanía Fátima Alcocer Espinosa

Dirigida por:
Otoniel Mario López Granado
Roberto Gutiérrez Mazón

D. OTONIEL MARIO LÓPEZ GRANADO, Profesor Contratado Doctor de la Universidad Miguel Hernández de Elche y D. ROBERTO GUTIÉRREZ MAZÓN, Profesor Contratado Doctor de la Universidad Miguel Hernández de Elche,

CERTIFICAN:

Que la presente memoria *Design and implementation of FPGA-based video encoding accelerators*, ha sido realizada bajo su dirección, en el Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica de la Universidad Miguel Hernández de Elche, por la Ingeniera Dña. Estefanía Fátima Alcocer Espinosa, y constituye su tesis para optar al grado de Doctora.

Para que conste, en cumplimiento de la legislación vigente, autorizan la presentación de la referida tesis doctoral ante la Comisión de Doctorado de la Universidad Miguel Hernández de Elche, firmando el presente certificado.

Elche, 12 de Junio de 2017

Fdo. D. Otoniel M. López Granado Fdo. D. Roberto Gutiérrez Mazón

D. PIEDAD NIEVES DE AZA MOYA, Catedrática de Universidad y directora del Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica de la Universidad Miguel Hernández de Elche,

CERTIFICA:

Que la presente memoria *Design and implementation of FPGA-based video encoding accelerators*, realizada bajo la dirección de D. OTONIEL MARIO LÓPEZ GRANADO y D. ROBERTO GUTIÉRREZ MAZÓN, en el Departamento de Ciencia de Materiales, Óptica y Tecnología Electrónica de la Universidad Miguel Hernández de Elche, por la Ingeniera Dña. Estefanía Fátima Alcocer Espinosa, constituye su tesis para optar al grado de Doctora.

Para que conste, en cumplimiento de la legislación vigente, autoriza la presentación de la referida tesis doctoral ante la Comisión de Doctorado de la Universidad Miguel Hernández de Elche, firmando el presente certificado.

Elche, 12 de Junio de 2017

Fdo. D. Piedad Nieves De Aza Moya

Acknowledgements

“Los dioses se han marchado, nos queda la televisión”.
Manuel Vázquez Montalbán

A mis personas favoritas en el mundo.

Gracias a mi familia. A mis padres, hermana y abuelos por ser los impulsores de todos mis sueños, donantes de inspiración. A Andrés por ser mi mitad y estar incondicionalmente a mi lado cuando más lo he necesitado. Sois mis personas favoritas en el mundo. Esto es por y para vosotros.

Gracias a mis directores de tesis. A Otoniel y Roberto porque nunca me han dejado sola durante este proceso, porque me han animado, ayudado y aguantado, y sobre todo, porque han demostrado ser mis amigos. También sois mis preferidos, no os cambiaría nunca.

Gracias a mis compañeros de GATCom. Al boss Mels, Pablo, Miguel, Héctor, Vicente y Oto again, por tratarme desde el primer día como una más, por las comilonas, partidas y risas que hemos compartido, porque os habéis convertido en unos amigos auténticos. Porque sois como de mi familia, concretamente como mis tíos, y a la familia hay que quererla.

Gracias a los compañeros de la Universidad de Gent que me acogieron con los brazos abiertos. A Jan, Glenn, Ruben, Niels, Tom y Johan por ser tan amables durante mi estancia.

Gracias a todos los que han escuchado la palabra tesis salir de mi boca, aunque fuera por un segundo. A mis tíos, amigas y medio Bigastro.

Mil gracias.

II

Abstract

Nowadays, having the latest image and video gadgets is trendy. Commercially, high performance multimedia devices are offered and/or demanded increasingly, such as very high-resolution TVs with high quality of image (Ultra High Definition (UHD)), video cameras that capture at very high frame rates, etc. Every day, millions of "selfies", "gifs", "boomerangs"' are immediately uploaded to social networks such as Instagram, Facebook or Twitter, and even we broadcast live our experiences with applications such as PeriScope. We are also spectators in first person of the most extreme sports which are recorded with Go-Pro cameras or similar. This is the reason why, in the field of image and video, many innovations and improvements will continue to be brought about due to the trend of high consumption on commercial multimedia devices.

Although nowadays we can find devices capable of reproducing and capturing very high resolution videos at high frame rates, this involves higher complexity in video data processing. Videos with previous requirements contain such a high amount of data that entails some problems such as the difficulty for video transmition in real time, the need for a large bandwidth that is almost unacceptable, the limited memory storage, and high power consumption, among others.

In order to overcome the above limitations, different coding standards have been developed over the years, trying to adapt to the market needs on each moment. As overview, video encoders compress the information so that it can be stored or transmitted occupying as little space as possible. As an example, platforms like Netflix use Google's VP9 compression codec to allow users download movies and series not needing too much storage space in order to watch them offline on any device. In order to achieve this compression, it is taken advantage of the huge redundancy of video sequences in both spatial and temporal domains. Thus, by removing such redundant information, it is possible to optimally encode video contents.

Therefore, due to the resource requirements, and the consequent increase in the complexity, and the processing time, this dissertation investigates the use of hardware accelerators based on Field Programmable Gate Array (FPGA)s on

the most complex parts that require more processing time in the video encoders.

Firstly, a hardware accelerator has been designed for the computation of the Motion Estimation (ME) of a High Efficiency Video Coding (HEVC) video encoder. In this case, the work has been focused on the latest video coding standard HEVC, which achieves the best compression efficiency in relation with its predecessors. As in previous standards, the removal of temporal redundancy demands an overwhelming computational cost, especially in high resolution video sequences. Therefore, the ME block (Inter prediction) of an encoder is one of the most critical modules in video compression. Based on this context, our design is based on the implementation of the ME block of a HEVC encoder on a FPGA, proposing two new and innovative techniques in both the tree adder for Sum of Absolute Differences (SAD) computation and the memory reading order. The results show that using our hardware ME module, a HEVC encoder is capable of encoding very high resolution video sequences faster than in real time.

Secondly, we present a FPGA-based hardware implementation of a very simple codec called Module Pulse Code Modulation (MPCM) based on the elimination of spatial redundancy (Intra prediction). This codec has the same advantages as Pulse Code Modulation (PCM) coding, reducing considerably the required bandwidth and maintaining the same image quality. The experimental results obtained demonstrate that our hardware implementation allows the continuous recording of a nowadays high-speed camera at a good quality image resolution.

This dissertation has been done under the modality of presentation of doctoral thesis with a set of publications, included in the regulations of the Miguel Hernández University of Elche. In compliance with these regulations, the publications that compose this dissertation have been included as an annex, and the sections corresponding to the research general description, the overall summary of the results obtained, and the final conclusions have been developed.

Resumen

Hoy en día poseer el último grito en dispositivos de vídeo e imagen no es nada sorprendente, de hecho, es la tendencia actual. Comercialmente, los dispositivos de altas prestaciones como televisores de muy alta resolución con una gran calidad de imagen o videocamáras que capturan a muy altas tasas de frames se ofertan y demandan cada vez más. Todos los días, se realizan millones de “selfies”, “gifs”, “boomerangs” que son inmediatamente subidos a redes sociales como Instagram, Facebook o Twitter; incluso transmitimos nuestra vida en directo con aplicaciones como PeriScope. También somo espectadores en primera persona de los deportes más extremos que son grabados en directo con cámaras del tipo Go-Pro. Es por ello, que el ámbito de la imagen y vídeo es un campo con futuro, en el cual cada día se aportan innumerables innovaciones y mejoras debido al consumo actual de dichos dispositivos comerciales.

Aunque cada vez es más fácil encontrar dispositivos capaces de reproducir y capturar vídeos de muy alta resolución a altas tasas de frame, estas altas prestaciones suponen una mayor complejidad en los procesos de tratamiento de vídeo debido a la gran cantidad de datos que contienen. En este contexto, nos encontramos con varios problemas como son la imposibilidad de transmitir vídeo en tiempo real ya que se necesitaría un ancho de banda inasumible y la dificultad de almacenamiento en memoria que siempre es limitada.

Con el fin de superar las limitaciones anteriores, se han desarrollado a lo largo de los años diferentes estándares de codificación de vídeo que tratan de adaptarse a las necesidades de cada momento. De manera muy general, los codificadores de vídeo comprimen la información para que pueda ser almacenada o transmitida ocupando el mínimo espacio posible. Como ejemplo, plataformas como Netflix utilizan el códec de compresión VP9 de Google para descargar películas y series que no ocupen demasiado y poder visualizarlas offline en cualquier dispositivo. Para conseguir dicha compresión, los codificadores aprovechan la alta redundancia de las secuencias de vídeo tanto en el dominio espacial como temporal, de manera que eliminando dicha información redundante, se consigue codificar de manera óptima el contenido de vídeo.

Por tanto, debido a la gran cantidad de recursos requeridos y el consecuente aumento en la complejidad y el tiempo de procesado, en esta tesis se investiga el uso de aceleradores hardware basados en FPGAs sobre las partes más complejas y que requieren más tiempo de procesado en los codificadores de vídeo.

En primer lugar, se ha diseñado un acelerador hardware para el cómputo de la estimación de movimento de un codificador de video HEVC. En este caso, el trabajo se ha centrado en el último estándar de codificación de vídeo HEVC, el cual muestra la mejor eficiencia de compresión respecto a sus predecesores. Al igual que en estándares anteriores, la eliminación de la redundancia temporal demanda un coste computacional abrumador, especialmente en secuencias de video de alta resolución. Por ello, el bloque de estimación de movimiento del codificador (predicción Inter) es uno de los módulos más críticos en la compresión de vídeo. Partiendo de este contexto, nuestro diseño se basa en la implementación de la estimación de movimiento de un codificador HEVC sobre FPGA, proponiendo dos técnicas novedosas, tanto en el árbol de sumadores para el cálculo de la estimación, como en el orden de lectura de memoria. Los resultados muestran que utilizando nuestro módulo hardware de estimación de movimiento, un codificador HEVC es capaz de codificar secuencias de muy alta resolución a tasas de frame más altas que las que se requieren a tiempo real.

En segundo lugar, se presenta una implentación hardware sobre FPGA de un codec muy sencillo llamado MPCM basado en la eliminación de la redundancia espacial (predicción Intra). Este codec presenta las mismas ventajas que la codificación PCM, reduciendo considerablemente el ancho de banda necesario y manteniendo la misma calidad de imagen. Los resultados experimentales obtenidos demuestran que nuestra implementación hardware permite la grabación continua a muy buena calidad en cámaras actuales de alta velocidad.

Esta tesis se ha realizado bajo la modalidad de presentación de tesis doctorales con un conjunto de publicaciones recogida en la normativa de la Universidad Miguel Hernández de Elche. En cumplimiento de dicha normativa se han incorporado las publicaciones que la componen como anexo y se han incluído las secciones correspondientes a la descripción general de la investigación, el resumen global de los resultados obtenidos y las conclusiones finales.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation

Activities carried out in this dissertation are directly related with the design of efficient image and video coders. These research activities are motivated by the interest and impact of the latest new video compression standards to meet the market multimedia trends, such as very high definition video content (resolutions 4K and 8K). Therefore, this work has been focused on the FPGA hardware implementations of image and video coders that are able to operate at very high-speeds with High Definition (HD) and UHD formats.

A video signal is represented as a sequence of frames of pixels. There exists a vast amount of redundant information that can be eliminated with video compression technology so that transmission and storage becomes more efficient. The similarity among different frames is called temporal redundancy, whereas the homogeneity inside single frames is know as spatial redundancy. Most video codecs use both spatial and temporal compression. In order to facilitate interoperability between compression at the video producing source and decompression at the consumption end, several generations of video coding standards have been defined and adapted. In this way, big companies use these video coding standards since they can give a significant technical and commercial edge to a product, by providing better image quality, greater reliability and/or more flexibility than competing solutions. At the moment, for low-end applications, software solutions are enough, but for high-end applications, dedicated hardware solutions are needed. [1]

Taking into account this scenario, the need to design hardware architectures that accelerate video coding arises in order to alleviate the computational complexity of current encoders, reaching high frame rates at ultra high resolutions. Therefore, now we will explain in a more detailed way, the most relevant aspects of both video coding and hardware design with FPGAs.

On the one hand, regarding video coding standards, video coding standards are mainly developed by two world organizations ISO/IEC MPEG and ITU-T VCEG [2, 3].

Three of the ITU-T VCEG video coding standards are the following:

- ITU-T H.120 *Codecs for videoconferencing using primary digital group transmission* was the first international standard for digital video compression. It was originally developed in 1984 and substantially revised in 1988, including such pioneering developments as motion-compensated inter-frame coding.

- ITU-T H.261 *Video codec for audiovisual services at p x 64 kbit/s* was the first commercially-successful digital video coding standard, and introduced the modern architecture of hybrid block-based video coding technology.

- ITU-T H.263 *Video coding for low bit rate communication* provided substantial improvements for real-time video coding communication, and was deployed in mobile devices as well as video conferencing systems.

Anothers video coding standards have been developed by ISO/IEC (also known as MPEG), such as:

- MPEG-1 *Coding reasonable quality images and sound at low bit rates* was established in 1992. MPEG-1 can be encoded at bit rates as high as 4-5Mbits/sec, but the strength of MPEG-1 is its high compression ratio with relatively high quality. MPEG-1 is also used to transmit video over digital telephone networks. MPEG-1 audio compression is more popularly known as MP3 and has revolutionized the digital music domain.

- MPEG-2 *Higher quality images at higher bit rates* was developed in 1994. MPEG-2 is the standard specified for DVD. The primary users of MPEG-2 are broadcast and cable companies who demand broadcast quality digital video and utilize satellite transponders and cable networks for delivery of cable television and direct broadcast satellite.

- MPEG-4 *Coding of audio-visual objects* was introduced in 1998. MPEG-4 include compression of Audio and Visual (AV) data for web (streaming media) and CD distribution, voice (telephone, videophone) and broadcast television applications. MPEG-4 absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, adding new features such as extended Virtual Reality Modeling Language (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. Initially, MPEG-4 was aimed primarily at low bit-rate video communications; however, its scope as a multimedia coding standard was later expanded. MPEG-4 is efficient across a variety of bit-rates ranging from a few kilobits per second to tens of megabits per second. MPEG-4 provides the following functions: improved coding efficiency over MPEG-2, ability to encode mixed media data (video, audio, speech), error resilience to enable robust transmission, and ability to interact with the audio-visual scene generated at the receiver.

- MPEG-7 *Multimedia Content Description Interface.* MPEG-7 is the latest proposal in the family of MPEG standards and will be formalized into a

standard by September 2000. MPEG-7 will be a standardized description of various types of multimedia information. MPEG-7 will not replace MPEG-1, MPEG-2 or MPEG-4. It is intended to provide complementary functionality to these other MPEG standards, representing information about the content, not the content itself (the bits about the bits). This functionality is the standardization of multimedia content descriptions.

Three other video coding have been developed collaboratively by both ISO/IEC MPEG and ITU-T VCEG:

- ITU-T H.262 — ISO/IEC 13818-2 *Generic coding of moving pictures and associated audio information.* Video is the result of the development under the collaborative team of the ITU-T advanced video coding rapporteur group and MPEG. It ushered in the era of digital television as it is known today.
- ITU-T H.264 — ISO/IEC 14496-10 *Advanced video coding for generic audiovisual services* is the result of the development under the collaborative team known as the JVT. It has become the dominant video coding technology world-wide and now accounts for roughly half of all communication network traffic world-wide (and over 80% of Internet video).
- ITU-T H.265 — ISO/IEC 23008-2 *High efficiency video coding (HEVC)* is the result of the development under the collaborative team known as the JCT-VC. It is now emerging as a substantial advance over prior designs to ease pressure on global networks and usher in an era of ultra-high definition television.

Any codec that is compatible with H.261/263/264/265 or MPEGs (1/2/4) has to implement a similar set of basic coding and decoding functions (although there are many differences of detail between the standards and their actual implementations). The model that these standards have in common, is often described as a hybrid DPCM/DCT codec.

Hybrid video coding is a scheme used by many video *codecs* (coder-decoder) that includes both *predictive coding* and *transform coding* in the compression and decompression processes. A hybrid video coding scheme consists of several stages. First, it makes use of the temporal or spatial redundancy present in a video sequence in order to *predict* a region of a frame, and this prediction is subtracted from the region that is currently being encoded. Then the residuum of this subtraction is *transformed* into the frequency domain and the resulting coefficients are quantized (lossy compression). Finally, the quantized coefficients are ordered and entropy coded.

In the encoding process, each frame is divided into small square regions or blocks. These blocks can be encoded using one of three modes: (a) without any prediction, (b) using spatial prediction, or (c) using temporal prediction. *Spatial prediction* exploits redundancy within a frame. In order to encode a block, it uses previously encoded regions of the same frame to search for pixel information that is similar to that block in order to create a candidate. Then, this candidate is subtracted from the current block and thus we obtain the residuum of the prediction. This method is also called *intra*-frame prediction. The article *MPCM: a hardware coder for super slow motion video sequences* shows the hardware design of an MPCM encoding that addresses the elimination of spatial redundancy in an image or frame in an efficient way to achieve very high frame rates.

Temporal prediction uses previously encoded frames to estimate a block candidate by means of the search of a similar block in other frames (called reference frames), which have been previously encoded, decoded, and stored in a buffer. It exploits temporal redundancy, taking advantage of the fact that nearby frames usually contain blocks that are very similar to the current block and so the residuum of the compensation is close to zero. This method is also called *inter*-frame prediction. The article *Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder* shows the implementation of the HEVC motion estimator block in hardware which deals with the inter-frame prediction.

On the other hand, we have used FPGA technology, since it encourages design reuse and can greatly enhance the upgradability of digital systems. A FPGA is a semiconductor device that can be programmed after manufacture to perform a specific application design, typically specified as a digital logic system [4]. They are being used for many real-life applications including communications, encryption, video image processing, medical imaging, network security and numerical computations. Specifically, the programmability of FPGAs is particularly useful for highly flexible encoding systems that can accommodate a multitude of existing standards as well as the emergence of new ones.

FPGAs can potentially approach the execution speed of application specific hardware with the rapid programming time of microprocessors. In recent years, the size of FPGAs has followed Moore's law: the number of logic gate doubles every 18 months. FPGAs can exploit improvements following Moore's law better than microprocessors because of their simpler and more regular structure.

The two major FPGA vendors are Altera and Xilinx. The fundamental building block of Xilinx FPGAs is the logic cell. In current Xilinx FPGA families, a logic cell comprises a 6-input Look-up table (LUT), two

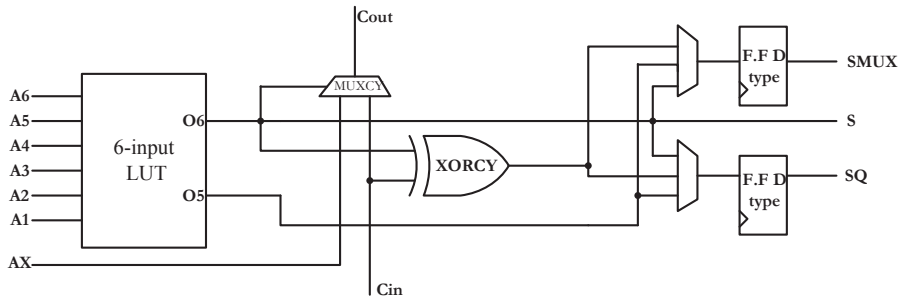


Figure 1.1. Simplified view of a Xilinx logic cell

multiplexers, and two registers. LUTs can be configured as either one 6-input LUT (64-bit Read Only Memorys (ROMs)) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a flip-flop. A simplified view of a logic cell is depicted in Figure 1.1.

Four LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a Configurable Logic Block (CLB). Four of the eight flip-flops per slice (one flip-flop per LUT) can optionally be configured as latches. Between 25-50% of all slices can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers (SRL32) or as two SRL16s. Modern synthesis tools take advantage of these highly efficient logic, arithmetic, and memory features. Futhermore, recent generation reconfigurable hardware has a large amount of resources, for instance, the Xilinx Virtex UltraScale XCVU440 has 5,065,920 Flip-Flops and 2,532,960 LUTs.

The architecture of a typical FPGA is illustrated in Figure 1.2. In general, an FPGA will have an array CLBs, programmable wires, and programmable switches to realize any function out of the logic blocks and implement any interconnection topology. Programming is done using of the many popular technologies such as SRAM cells, antifuses, EPROM transistors and EEPROM transistors. In addition to logic blocks, nowadays FPGAs such as Xilinx Virtex-7 devices contain embedded hardware elements for memory, multiplication, multiply-and-add and a number of hard microprocessor cores (such as ARM Cortex-9 or the IBM PowerPC), and even, they are part of a System-On-Chip (SoC) embedded with external memory and hard peripherals such as Xilinx Zynq-7000 families.

The long IC fabrication time is completely eliminated for these devices and design realization times are only a few hours. The idea of

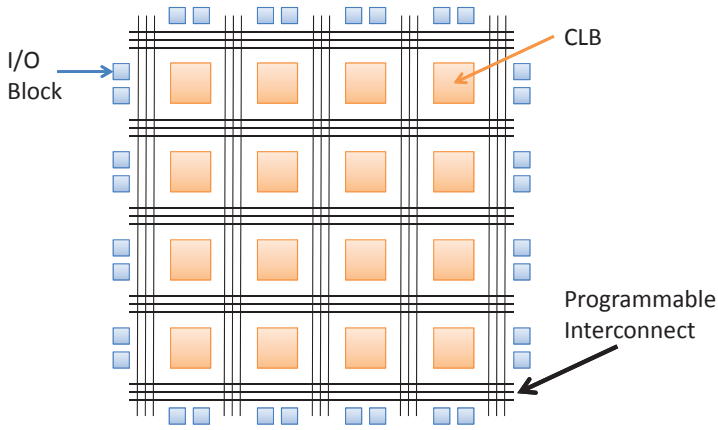


Figure 1.2. Architecture of a typical FPGA

user-programmability is very exciting, most ASIC vendors now prefer FPGAs for low cost prototyping for fine tuning of designs before fabrication. Also, from a marketing point of view, the FPGA technology allows quick product announcements, which is commercially attractive [5].

During the development of this thesis, several FPGAs and SoCs of Xilinx have been used. Xilinx devices are now much larger and come with a variety of new technology, including 144 kb UltraRAM, DSP48E2, high-speed transceivers up to 28 Gigabits (Gbs) Input-Output (I/O) interfaces, hardened microprocessors and peripherals, analog mixed signal, and more. These larger and more complex devices create multidimensional design challenges, but handled correctly, they can achieve faster time-to-market and increase productivity.

1.2 Objectives

The general objectives of this thesis have been the following:

- Design of efficient hardware image and video encoders.

- Develop prototypes of FPGA accelerators that allows high-speed video coding at high-definition formats.

Based on the previous general objectives, the specific objectives of this work are detailed below:

- Study, design and implementation of a fast and simple hardware encoder that allows continuous capture and coding in real time with ultra high-speed cameras.

- Analysis of the computational complexity of nowadays video encoders.

- Design and implementation of a hardware integer motion estimation module for HEVC video encoder. This new hardware module will speed up its coding process and eliminate its high computational cost.

- Study and evaluation of the balance between the throughput, compression rates, image quality, the hardware complexity and the FPGA usage.

1.3 Articles

This dissertation has been carried out in the modality of a doctoral thesis presented with a set of publications. According to the Internal Regulations of the Miguel Hernandez University for the presentation of doctoral theses with a set of publications, this dissertation includes a general introduction where the work done is presented and justified. It also incorporates a global summary with the results obtained, their discussion and the final conclusions, in both languages English and Spanish. Finally, an annex with the publications presented in their original language is attached.

The publications presented in this thesis are:

- **Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder**
  Estefania Alcocer, Roberto Gutierrez, Otoniel Lopez-Granado, and Manuel P. Malumbres
  Journal of Real-Time Image Processing. Springer Verlag Berlin Heidelberg 2016
  Impact factor: 1.564
  Category Name: ENGINEERING, ELECTRICAL & ELECTRONIC
  Quartile in Category: **Q2**

ISSN 1861-8219
http://dx.doi.org/10.1007/s11554-016-0572-4

- **MPCM: a hardware coder for super slow motion video sequences**
  Estefania Alcocer, Otoniel Lopez-Granado, Roberto Gutierrez, and Manuel P. Malumbres
  EURASIP Journal on Advances in Signal Processing 2013. Springer Open Journal
  Impact factor: 0.808
  Category Name: ENGINEERING, ELECTRICAL & ELECTRONIC
  Quartile in Category: **Q3**
  ISSN 1687-6180
  http://dx.doi.org/10.1186/1687-6180-2013-142

The publications explained below focus on the design of hardware systems that perform the computationally costliest parts of a codec in both video and image video coding in order to achive ultra-fast encoding at high frame rates for high-resolution formats.

On the one hand, the first publication deals with hardware Integer Motion Estimation (IME) for an HEVC video encoder. HEVC standard is the most recent joint video project of the ITU-T VCEG and ISO/IEC MPEG standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [6]. HEVC has been designed to cope with new video services, working with higher video resolutions and adapting its design to allow the use of parallel processing techniques. It can compress video about twice as much as its predecessor, H264/Advanced Video Coding (AVC), without sacrificing quality, but increasing the computational complexity. As in previous standards, ME is one of the encoder critical blocks to achieve significant compression gains, but it demands an overwhelming complexity cost to accurately remove video temporal redundancy. The purpose of this work is the implementation of a HEVC ME block in hardware in order to reduce the overall video encoding time and to achieve real-time encoding for high-resolution videos.

On the other hand, the second article is based on the hardware implementation of a very simple codec called MPCM, whose coding/decoding process of each pixel is based on the prediction and interpolation of neighboring pixels of the same frame. The purpose of this work is to take advantage of a codification as simple as possible to capture high resolution videos at ultra high frame rates.

1.3.1 *Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder*

In this work, a design of a new hardware architecture which perform IME computation in a fast and accurate way is proposed, in order to significantly reduce the computation cost of the overall encoder.

The ME technique is based on the similarity between adjacent video frames, predicting the current frame based on a previous or subsequent reference frame in order of appearance. The Motion Vector (MV) represents the translational movement of a picture area in the current frame compared to its position in the reference frame.

In the HEVC video coding standard, each frame is partitioned into Coding Tree Units (CTUs) which cover a rectangular picture area of *LxL* samples. The value of *L* may be equal to 16, 32, or 64 as determined by an encoded syntax element specified in the Sequence Parameter Set (SPS). The larger size typically enables better compression, particularly beneficial when encoding high-resolution video content. The CTU is the basic processing unit used in the standard that can be directly used Coding Units (CUs) or can be further partitioned into multiple CUs. The partitioning is achieved using tree structures. The CTU contains a quadtree syntax that allows for splitting the CUs to a selected appropriate size based on the signal characteristics of the region that is covered by the CTU. The quadtree splitting process can be iterated until the size reaches the minimum allowed that is selected by the encoder using syntax in the SPS and is always 8x8 samples or larger. The prediction mode for the CU is signaled as being intra or inter, according to whether it uses intra-picture (spatial) prediction or inter-picture (temporal) prediction. When the prediction mode is signaled as intra, the CU can be split into four quadrants that each have their own intra prediction mode, from the CU size until 4x4 blocks. When the prediction mode is signaled as inter, that is the case which concerns us, it is specified whether CUs are split into one, two, or four prediction units. When a CU is split into four prediction units, each one covers a quadrant of the CU. When a CU is split into two prediction units, various types of this splitting are possible. The partitioning possibilities for inter-predicted CUs are depicted in Figure 1.3.

The first four partitions illustrate the cases of not splitting the CU of size 2Nx2N, of splitting the CU into two prediction units of size 2NxN or Nx2N, or splitting it into four of size NxN. The next four partition types in Figure 1.3 are referred to as Asymmetric Motion Partitions (AMPs). One prediction unit of the asymmetric partition has the height or width 2N/4 and width or height 2N, respectively, and the other prediction unit fills the rest of the CU by

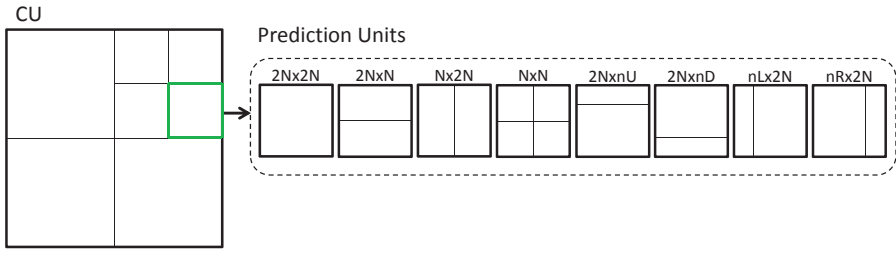


Figure 1.3. CU splitting modes into Prediction Units in inter prediction

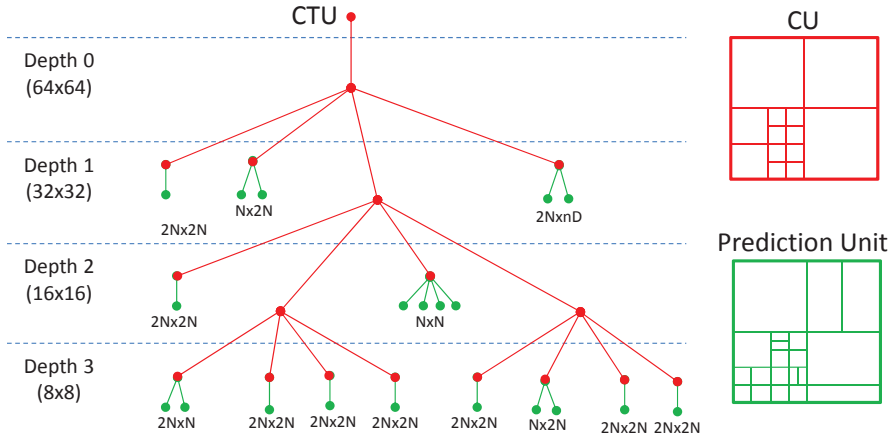


Figure 1.4. HEVC quad-tree structure: CUs and Prediction Units relationship

having a height or width of 3x(2N)/4 and width or height 2N. Each inter-coded prediction unit is assigned one or two motion vectors and reference picture indices. To minimize worst-case memory bandwidth, the minimum prediction unit size is restricted to 4x8 or 8x4. The quadtree syntax of the CTU, in which CUs and prediction units relationship is shown, is represented in Figure 1.4. In addition, as a real example, the Figure 1.5 shows the partitioning that is performed in a frame of the video sequence RaceHorses during the HEVC Motion Estimation.

Finally, in the HEVC inter-prediction process, the total number of different partitions for a 64x64 CTU is more than 600, and for each of these partitions, the HEVC encoder performs one ME process to determine the best CU partitions in terms of bit rate and video quality. In this way, in this publication a high-performance IME hardware unit in HEVC that provides the minimum SADs and associated MVs of all possible partitions from a 64x64 CTU for inter-prediction is presented, exploiting parallelism in an efficient way, by

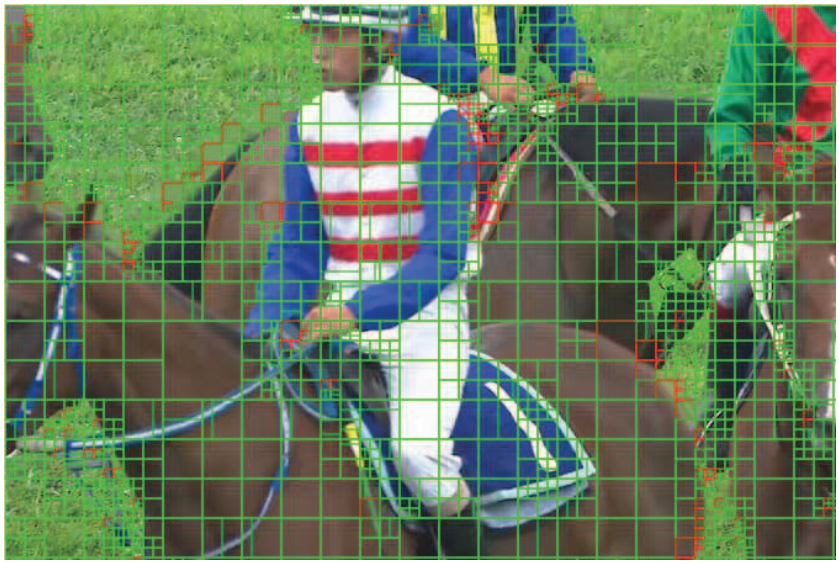


Figure 1.5. HEVC Prediction Units in a frame

means of a Full Search (FS) algorithm which exhaustively finds motion for all prediction unit blocks at every single point of the established search area since provides computational regularity and excellent video quality. In order to perform this implementation, the Xilinx FPGA Virtex-7 was used.

Focusing on the computation blocks of the hardware ME proposal, we present both innovative techniques: a new SAD adder tree structure, and a new memory scan order. Firstly, we designed a new SAD adder tree structure to perform the additions at the first level of the tree, starting from the maximum size of the CTU, and halving the amount of additions at the next tree levels. With our proposal, we took advantage of the resources provided by the FPGA, obtaining the minimum possible latency when calculating SADs of all levels and partitions for a CTU. Secondly, regarding the new memory scan order, a series of reconfigurable shift registers and processing elements are responsible for storing the necessary pixels of both reference and current frames, keeping them always available for calculating the SADs and MVs of a CTU. With our system, we avoid external memory accesses since available data are highly reused by reconfiguring the displacement in a more efficient way.

The architecture described above was modeled in VHDL, and was synthesized, simulated, and implemented on the Xilinx FPGA, Virtex-7 XC7VX550T-3FFG1158 [7]. The experiments were performed using different configurations of both search area sizes (128x128, 104x104, 64x64, 52x52,

and 32x32) and CTU sizes (64x64 and 32x32). The correctness of this design was tested and verified with the HEVC HM 14 reference model [8] working with the main profile and low-delay configuration mode, using three video sequences from the HEVC common conditions video set with different resolutions and frame rates.

First, the throughput for different configurations of CTU and search area sizes in the Virtex 7 FPGA technology was assessed, showing frequency (clock), latency, and the system throughput in terms of the maximum frame rate under different video formats (1080p, 2K, and 4K). For these configurations, the resources used on the Virtex-7 FPGA were also shown. Futhermore, our hardware design was compared with other state-of-art architectures which perform the same functionality under FPGA technology. Finally, the integration of our IME FPGA-based accelerator was analyzed in terms of speed-up and was observed how the different CTU and search area size configurations impact on the R/D performance of the HEVC encoder, with regard to the HEVC reference software. The results of this research are detailed in section 2.2.1.

1.3.2 *MPCM: a hardware coder for super slow motion video sequences*

In this work, a fast FPGA implementation of a simple codec called MPCM is proposed. This hardware codec is intended to allow current high-speed cameras to capture in a continuous manner.

Source coding algorithms must be extremely simple for new comercial high-speed cameras that capture high-resolution videos at frame rates up to 10,000 frames/second. Most of ultahigh-speed cameras store the captured images in a fast Synchronous Dynamic Random Access Memory (SDRAM) module of up to 64 GB [9, 10, 11, 12]. This approach of using fast SDRAM memory as video storage is feasible since the memory bandwidth is high enough, but when memory is run out, the camera stops recording and needs to save the stored video to a secondary storage in raw or compressed format. This is a limitation because depending on the capturing resolution of the camera, only a few seconds could be recorded in the Random Access Memory (RAM) module, and so continuous capturing is not possible. For these reasons, many times high-resolution videos at high frame rates are recorded or transmitted using PCM (raw video) since is the simplest technique [13]. PCM coding presents several advantageous properties, for instance, direct processing, random access, and rate scalability. However, the high bit rate of raw video can make its transmission or storage difficult. The huge

amount of data of the resulting uncompressed image/video needs to be processed to guarantee its transmission or storage, being a really challenging task. Thus, the internal communication bus may not be fast enough to transfer the video out of the camera, or the writing speed of the storage device may not be high enough to save the video [14]. So as to overcome these restrictions, it would be of interest both, to reduce the video storage requirements by means of hardware encoders that fulfill the application requirements such as high frame rate and ultra-high-definition video formats, and to consider a coding algorithm which had properties similar to those of PCM (low complexity, random access, and scalability) with a better coding efficiency.

MPCM [15] image coder is able to reduce the rate of the PCM signal in a very simple way without losing the advantages provided by PCM coding, as prososed in [16]. In this algorithm, MPCM encoder removes certain bits from each pixel value which represents a very simple processing, in order to encode an image. At the decoder side, where the bits that were removed from each pixel will be predicted by using its codeword (remaining bits of a pixel) and Side Information (SI) that the decoder computes by interpolating the previously decoded pixels. In this work, a hardware codec based on MPCM on a XC7Z020-1CLG484CES Xilinx FPGA device was implemented. Both encoder and decoder architecture designs were tested and validated.

Regarding results, an evaluation of the complete system was performed in terms of Peak Signal-to-Noise Ratio (PSNR), encoding/decoding times, board area usage, maximum frame rate, and resulting speed-ups when compared to a CPU sequential algorithm. The results of this research are detailed in section 2.2.2.

Chapter 2

Research results

Contents

2.1 Materials and methods

The resources described below are those used to carry out the research of this thesis:

- **Vivado Design Suite of Xilinx**
  This tool suite is intended to increase the overall productivity for designing, integrating, and implementing systems using the Xilinx UltraScale and 7 series devices, Zynq UltraScale + MPSoC device, and Zynq-7000 All Programmable (AP) SoC [17]. It includes place and route tools that analytically optimize multiple and concurrent design metrics, such as timing, congestion, total wire length, utilization and power. It replaces all of the ISE Design Suite point tools. Vivado introduces the concept of opening designs in memory. Opening a design effectively loads the design netlist at that particular stage of the design flow, assigns the constraints to the design, and applies the design to the target device. This allows to visualize and interact with the design at each design stage and it enables to open designs after Register-Transfer Level (RTL) elaboration, synthesis, and implementation. It can be made changes to constraints, logic or device configuration, and implementation results, as well using design checkpoints to save the current state of any design. A design checkpoint is a snapshot of the design at any stage of the design process that includes the netlist, constraints, and implementation results. Vivado automatically creates design checkpoints at each stage of the flow that can be opened and analyzed.

- **ISE Design Suite 14 of Xilinx**
  This tool is a predecessor development environment of Vivado Design Suite. Now, Vivavo has replaced this solution. With performances lower than those currently provided by Vivado, ISE supplies a complete solution for logic and connectivity design encompassing the front-to-back base methodology and Intellectual Property (IP).

- **HEVC reference software**
  The reference software for HEVC is called HEVC Test Model (HM). This software has been used for the coding of video sequences, specifically the HM-14 version [8]. In the software repository, the configuration files are also included, which have been modified to obtain the different configurations we needed in this work (search algorithm, block sizes, search area, etc.)

- **Matlab/Simulink**
  Matlab is a program widely used in engineering to perform technical,

scientific and general purpose calculations. It integrates calculation operations, visualization and programming. In the case of this research, Matlab, with the help of the Simulink application, has been used for the development, modeling, simulation and prototyping of the mathematical algorithms that have been implemented in both image and video coding, in order to verify the reliability of the model implemented in hardware.

- **Software platform**
  All software tests have been run over an Intel Core i7-3770 CPU 3.40 GHz with 16 GB RAM.

- **Video sequences**
  The video sequences used for coding shown at Table 2.1 have been selected from the HEVC common conditions [18].

Table 2.1. Video sequences used during the research

Video sequence	Resolution (pixels)	Frame-rate (fps)
Racehorses	832x480	30
ParkScene	1920x1080	24
Basketball Drive	1920x1080	50
People On Street	2560x1600	30
Traffic	2560x1600	30

- **Images**
  In Table 2.2 we find the set of gray-scale images that have been used in this work, in order to test the image codec. In all of them, each pixel is represented with 8 bits.

- **Virtex-7 FPGA of Xilinx**
  Virtex-7 FPGAs are optimized for system performance and integration at 28nm and bring best-in-class performance/watt fabric, DSP performance, and I/O bandwidth to your designs. The family is used in an array of applications such as 10G to 100G networking, portable radar, and ASIC Prototyping [7]. At the present work, the XC7VX550T-3FFG1158 device has been used.

- **Zynq-7000 SoC 7Z020 of Xilinx**
  This product integrates a feature-rich dual-core ARM Cortex-A9 based Processing System (PS) and 28 nm Xilinx Programmable Logic (PL) in a single device. The FPGA included in this device is the

Table 2.2. Images used during the research

Image	Resolution (pixels)
Zelda	512x512
Lena	512x512
Peppers	512x512
Barbara	512x512
Baboon	512x512
Tractor	1920x1080
Woman	2048x2560
Ducks	3840x2160

XC7Z020-1CLG484CES. The ARM Cortex-A9 CPUs are the heart of the PS and includes 1GB DDR3 memory, 16MB Quad SPI Flash, HDMI Video OUT and a rich set of peripheral connectivity interfaces [19].

- **Avnet Zynq-7000 SoC Mini-ITX of Xilinx**
  It is a complete development platform for designing and verifying applications based on the Xilinx Zynq-7000 All Programmable SoC family. In addition to the Xilinx Zynq-7000 AP SoC XC7Z100 device (XC7Z100-2FFG900), the Zynq Mini-ITX development board features 2 GB DDR3 SDRAM, PCIe Gen2 x16 Root Complex slot (x4 electrical), SATA-III interface, SFP interface, QSPI Flash memory, HDMI interface, LVDS touch panel interface, Audio Codec, a 10/100/1000 Ethernet PHY, a USB 2.0 4-port hub, a microSD card interface, and a USB-UART port [20].

The main features of the 3 FPGAs used (Virtex-7 and the corresponding ones to the two evaluation boards) are detailed in the Table 2.3

Table 2.3. FPGAs Feature Summary

FPGA	Virtex-7 XC7VX550T	Z-7020 XC7Z020	Z-7100 XC7Z100
LUTs	346400	53200	277400
Flip-flops	692800	106400	554800
BRAM (36Kb Blocks)	1180	140	755
DSP	2880	220	2020
Speed Grade	-3	-1	-2

2.2 Results

In this section an overall summary of the results obtained from research on both video and image accelerators has been presented. Implementations of these accelerators have been analyzed in terms of R/D, processing time, board area usage, maximum frame rate, and speed-ups when compared to software algorithms.

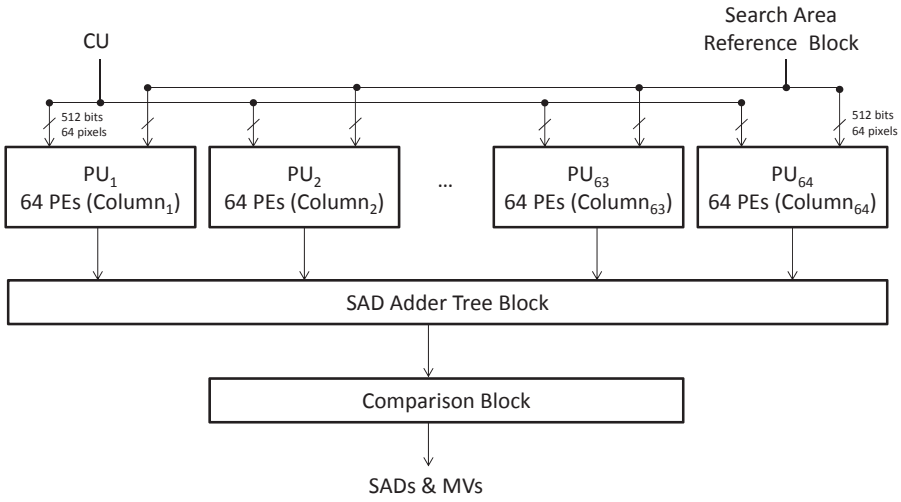
2.2.1 Video coding acceleration

As described in Section 1.3.1, a high-performance IME hardware unit in HEVC that provides the minimum SADs and associated MVs has been implemented. The architecture is based on both a new memory scan order and a new adder tree structure, which supports all partitioning modes and allows different configurations, such as (a) the maximum CTU size with values of 64x64 and 32x32, and (b) the size of the search area of the reference frame with values defined as the double size of the CTU, 80% of the double size of the CTU, and the same size as a CTU.

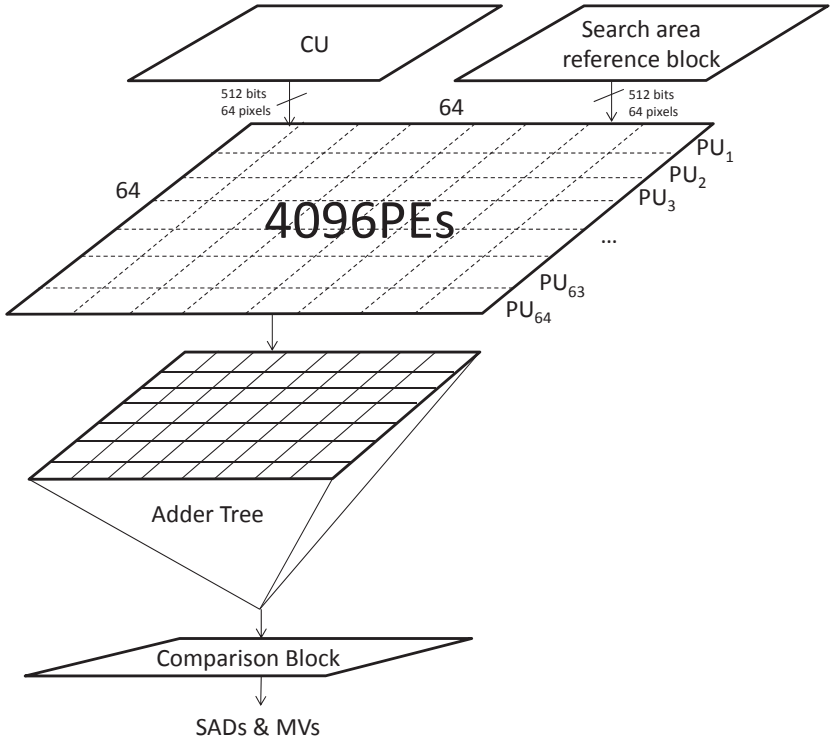
The system is composed of memory areas for current CU and reference search area pixels, 64 Processing Units (PUs), one SAD Adder Tree Block (SATB), and one comparison block that saves the minimum SAD values and their corresponding MVs for all CU partitions. Figure 2.1 shows the structure of hardware IME architecture.

Regarding our new innovative technique about memory scan order, a snake scan order and a reconfigurable data path with 64 propagation registers were adopted in order to provide high data reuse. The snake scan order visits all positions of the search area following a Hamiltonian path composed by consecutive vertical scans with alternating scan directions (the first vertical scan begins from top to bottom, then moves one pixel to the right and starts the next vertical scan in a bottom to up direction, and so on) as illustrated in Figure 2.2.

There are three scanning directions U (upward), D (downward), and R (rightward). The current 64x64 CTU pixels are stored in the Processing Element (PE)s only once (at the beginning). The reference pixels will also be loaded to the PEs but instead of loading from Block Random Access Memory (BRAM), where the pixels belonging to a reference frame area are stored, will be loaded from the shift registers, since they will help us to perform the snake scan order and as a consequence a huge reduction of memory load operations will be achieved.



(a) Hardware blocks



(b) SAD unit concept

Figure 2.1. General structure IME architecture

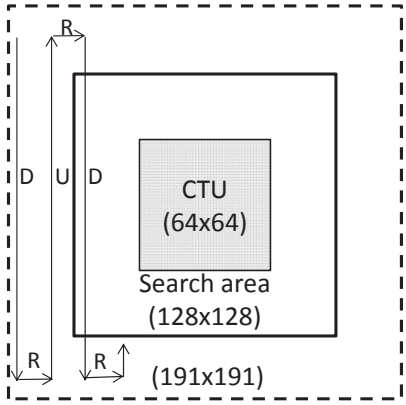


Figure 2.2. Scan order of the search area

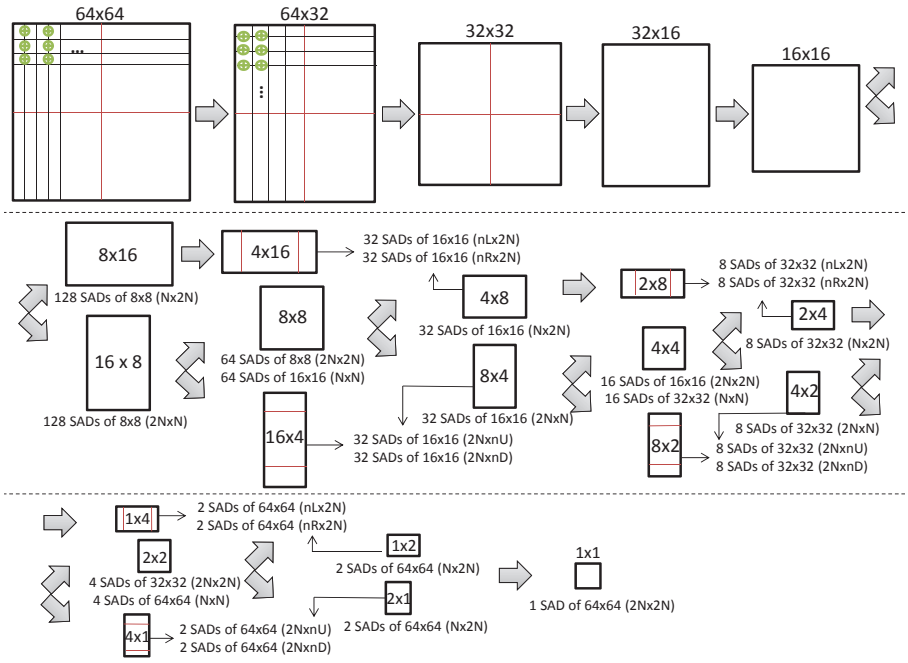


Figure 2.3. Structure of SAD Tree Block

Regarding the second new method, a SATB block which computes the SAD values for all partitions of each 64x64 CTU at every clock cycle has been developed. After receiving the 64x64 distortions from PUs associated to the current search area position, a succession of aggregation stages are performed in this block to compute the corresponding SAD values for all the CTU partitions (a total number of 677), as shown in Figure 2.3.

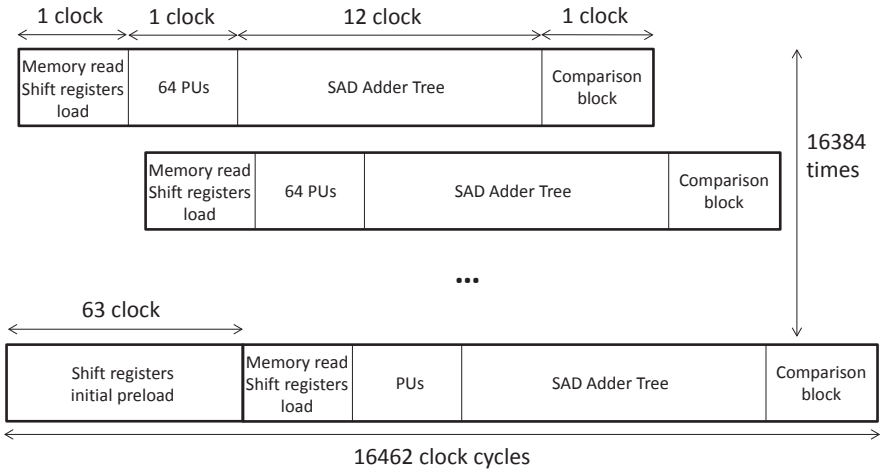


Figure 2.4. Pipeline process of the proposed architecture

Table 2.4. Throughput for different configurations in Virtex-7

CTU size	64x64	64x64	64x64	32x32	32x32	32x32
Search area	128x128	104x104	64x64	64x64	52x52	32x32
Clock (MHz)	247	247	247	318	318	318
Latency	16462	10894	4174	4142	2750	1070
Fps at 1080p	32	48	124	39	59	151
Fps at 2K	30	45	116	37	55	141
Fps at 4K	8	12	30	10	15	37

At each stage, all pairs of consecutive columns/rows are added, reducing to half the width/height of the resulting partition. This SAD aggregation process is followed until the last partition size is reached (1x1), ie. the SAD corresponding to the 64x64 partition. At intermediate stages, the SADs of the rest of partitions are stored. Finally, in the proposed architecture, the SATB module delivers 677 SADs of the current CTU block every single clock cycle to the next module, the comparison block.

This work was modeled in VHDL, and it was synthesized and simulated on the Xilinx FPGA, Virtex-7 XC7VX550T-3FFG1158. In Figure 2.4, and Tables 2.4, 2.5 and 2.6 the pipeline process with the total latency, the throughput, and the resources used for different configurations are shown, repectively, when our hardware architecture is implemented on the Virtex-7 FPGA described above.

Table 2.5. Utilization resources for 64x64 CTU implementation in Virtex-7

Resources	Flip-Flops	LUTs	Memory (kB)
Memory Read Controller Block	36657 (25.40%)	36413 (19.30%)	36 (100%)
PUs (Distortion computation)	32768 (22.71%)	94208 (49.93%)	-
SAD Adder Tree Block (SATB)	58727 (40.70%)	47063 (24.95%)	-
Comparison Block	16150 (11.19%)	10980 (5.82%)	-
TOTAL	144302	188664	36

Table 2.6. Utilization resources for 32x32 CTU implementation in Virtex-7

Resources	Flip-Flops	LUTs	Memory (kB)
Memory Read Controller Block	10155 (27.55%)	9812 (20.22%)	9 (100%)
PUs (Distortion computation)	8192 (22.22%)	24541 (50.57%)	-
SAD Adder Tree Block (SATB)	14580 (39.55%)	11445 (23.58%)	-
Comparison Block	3937 (10.68%)	2733 (5.63%)	-
TOTAL	36864	48531	9

As shown, in our architecture, with a configuration of a 64x64 CTU size and a search area of 128x128 pixels, the memory reading process and shift registers propagation require only one clock cycle. The PUs use one cycle, the SATB requires twelve additional clock cycles, and the comparison block needs one additional clock cycle. So, the proposed architecture requires 63 clock cycles to perform the initial load of the shift registers, 15 clock cycles to load the pipeline, and then as many clock cycles as positions the search area has. Finally, te whole process is done in 16462 clock cycles, that is, 63 clock cycles for the initial load, plus 15 clock cycles for the full processing in one pixel position, plus 16384 times to repeat the whole process of motion estimation (search window 128x128 pixels = 16384 pixels), with only one delay clock cycle in the pipeline. As can be seen in Table 2.4, our design can operate at the frequency of 247 and 318 MHz for a 64x64 CTU and a 32x32 CTU, respectively. Regarding the configuration with maximum sizes, whose latency is 16462 clock cycles, and taking into account the frequency obtained, the encoder carry out the IME process in 66.65μ seconds, obtaining a throughput of 30 frames per second (fps) at 2K video formats (2K@30fps). The encoder is able to process video in real time for both 1080p and 2K resolutions in all tested configurations, and also with 4K video formats if the search area size is the same as the CTU size. In terms of resources used, the

64x64 CTU size requires near four times more resources, which implies that the use of more resources in the design provides higher throughput, in this case, with a 4:3 relationship, as shown in Tables 2.5 and 2.6.

Futhermore, in Table 2.7, our hardware architecture is compared with other comparable state-of-art architectures implemented on different FPGA platforms for both the 64x64 CTU and the 32x32 CTU size, and different search area sizes.

Table 2.7. Comparison of the proposed architecture with state-of-the-art works

Design	Medhat [21]	Proposal 1	D'huys [22]	Proposal 2	Yuan [23]	Proposal 3
CTU size	64x64	64x64	64x64	64x64	32x32	32x32
Search area	104x104	104x104	64x64	64x64	48x48	48x48
Technology	Virtex-7	Virtex-7	Virtex-5	Virtex-5	Virtex-6	Virtex-6
Clock (MHz)	458.7	247	150	159	110	200
AMP	No	Yes	No	Yes	Yes	Yes
Throughput	2K@30fps	2K@45fps	720p@57fps	720p@173fps	1080p@30fps	1080p@43fps
Flip-Flops	39901	144302	199682	178620	19744	43531
LUTs	24957	188664	210158	184288	55346	45752
Memory (kB)	44	36	1229	36	148	9

Regarding results for the 64x64 CTU size, Medhat et al. [21] present a parallel SAD block for the HEVC integer-pel FS architecture without supporting AMP modes with a search area of 104x104 pixels. They used the Virtex-7 technology, and their design can operate at the frequency of 458.7 MHz. The operating frequency of our design with the same technology and configurations is almost two times lower. However, our architecture is capable of processing 45 fps at 2K video formats instead of 30 fps as obtained by the proposed design in [21]. Therefore, our proposed architecture is 1.5x as fast as the one proposed in [21] using the same search area size and considering all the AMP partition modes, contrary to [21], where AMP partitions are not calculated. This is due to the fact that our design takes advantage of the minimal latency to perform the same operations as we have an efficient pipeline design. Therefore, our system achieves higher throughput, reaching real-time processing for 2K video resolutions at 45 fps, and being on the way to accomplishing the same goal for 4K video formats, where 12 fps were obtained.

On the other hand, D'huys [22] proposes a reconfigurable design for HEVC motion estimation which can operate at the frequency of 150 MHz. His

architecture is compared with our proposal, setting a common search area size to 64x64 pixels and the Virtex-5 technology. The operation frequency of our proposal is 159 MHz, achieving system throughput of 20 fps at 4K and 75 fps at 2K video formats. Our design significantly improves the performance of the architecture presented in [22], which is able to process a lower resolution video (720p) at 57 fps. If the video resolution is set to 720p, our architecture is capable of processing 173 fps. So, our architecture presents good balance between the maximum frequency and pipeline processing design, taking advantage of the low latency by leveraging all available resources.

Regarding results for the 32x32 CTU size, we show the comparison results between our proposal (implemented on a Virtex-6 FPGA) and the IME design found in [23], both with a search area size of 48x48 pixels. The most significant feature, worthy of attention, is that our proposal can provide a higher operation frequency, achieving throughput of 43 fps at 1080p and 40 fps at 2K resolution, whereas the architecture presented in [23] is able to achieve 30 fps at 1080p video formats, using a similar amount of FPGA resources.

Considering the presented results, our architecture shows an efficient implementation of available resources in FPGA, overcoming the performance of previous state-of-the-art architectures.

Table 2.8. Time profile of the IME HEVC for a 64x64 CTU with video sequences *Race Horses*(s1), *Basketball Drive*(s2), and *People On Street*(s3)

Search area	128x128			104x104			64x64		
Video sequences	s1	s2	s3	s1	s2	s3	s1	s2	s3
Encoding Time SW (s)	13670	61602	135970	9392	42050	92863	4117	17881	39933
% IME Time SW	95	96	95	90	94	93	83	86	85
CTU/s SW	0.24	0.26	0.23	0.38	0.39	0.35	0.91	1.00	0.89
CTU/s HW	14993	14993	14993	22625	22625	22625	59172	59172	59172
HW gain	62260	57767	64800	59621	58312	65384	64856	59115	66477

Regarding the time profile of the HEVC IME module, a comparison between software and hardware executions has been performed. Tables 2.8 and 2.9 show the following information for different configurations allowed: The total time required to encode 10 frames of each video sequence with the software reference model, the percentage of the total time needed by the IME software module using a FS algorithm, the number of CTUs per second that can be computed by software and by hardware versions of the IME module, and finally, the gain obtained with the inclusion of our IME hardware module instead of the software one. These values depend on the size configurations,

Table 2.9. Time profile of the IME HEVC for a 32x32 CTU with video sequences *Race Horses*(s1), *Basketball Drive*(s2), and *People On Street*(s3)

Search area	64x64			52x52			32x32		
Video sequences	s1	s2	s3	s1	s2	s3	s1	s2	s3
Encoding Time SW (s)	3652	15892	35295	2634	11303	25293	1378	5748	12911
% IME Time SW	85	87	86	79	81	81	60	63	62
CTU/s SW	4	5	4	6	7	6	14	17	15
CTU/s HW	76923	76923	76923	115607	115607	115607	297619	297619	297619
HW gain	20383	17293	19457	20654	17384	19675	21096	17664	19912

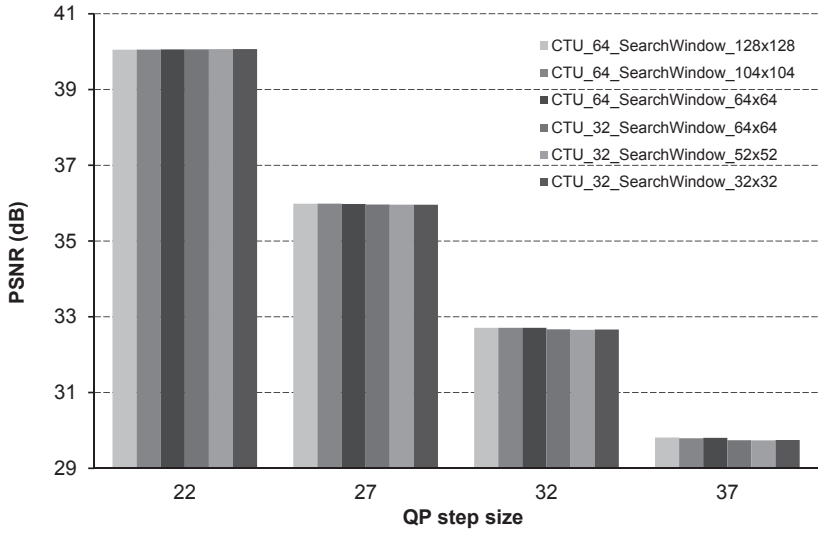


Figure 2.5. R/D performance for different CTU and search area sizes for the RaceHorses sequence

and in the case of the IME software module, also depend on the video sequence. Regarding these results, it could be conclude that the IME module is a bottleneck in the HEVC reference software, and if the IME software module is replaced by our FPGA-based device, the overall encoding time will be significantly reduced. In order to reduce the hardware complexity, allowing faster versions with reduced power consumption, the CTU size and the search area must be reduced as much as possible, without causing degradation in the encoding process, neither decreasing the overall video quality nor reducing the compression rate, as can be seen in Figure 2.5. In Figure 2.5, we show the video quality of the test video sequence RaceHorses (s1) for each CTU and

Table 2.10. Average CTU IME time with 2xCTU search area size

CTU size	Full search SW	Diamond search SW	Full search HW
64x64	4.11 s	4.65E-02 s	6.67E-05 s
32x32	2.48E-01 s	3.05E-03 s	1.30E-05 s

search area sizes at different compression levels (QP values). As can be seen, there are negligible differences in terms of R/D between the CTU size and search area size. Although R/D differences may depend on the video content, similar results were obtained for the other two video sequences tested.

The Diamond Search (DS) algorithm is used by default in the HM reference software due to this is about 90 times as fast as the FS algorithm in software, with the disadvantage that it does not guarantee finding optimal MVs, and as consequence video quality could be affected. Finally, as can be seen in Table 2.10, the inclusion of our IME hardware module, using FS algorithm, will speed up the IME computation of diamond-like search algorithm 230 times and 700 times for 32x32 and 64x64 CTU sizes, respectively. Therefore, as conclusion, the use of our IME hardware accelerator designed in a FPGA platforms are mandatory when real-time UHD video encoding is the objective.

2.2.1.1 Prototype implementation

Following the same research line about video coding acceleration, the previous hardware HEVC IME design has been evaluated when applied to a SoC platform. In this case, the Xilinx SoC, Zynq-7 Mini-ITX Motherboard XC7Z100 (xc7z100ffg900-2) has been used, as hardware platform.

A SoC consist of two well-defined parts, a PS based on an ARM processor and several hard peripherals like Ethernet, USB, etc., and a PL been made up of a FPGA. In Figure 2.6 the top-level hardware architecture is showed, where an ARM processor manages the transfer between the IME SAD module and a Double Data Rate (DDR) memory which stores both reference and current frames, by a Direct Memory Access (DMA) module. With this hardware platform, ARM processor works at 666.66 MHz, and the DDR at 533.33 MHz, whereas the clock frequency of the PL is restricted by the maximum frequency of the SAD HEVC module which is the responsible for the IME calculation. In the case of 64x64 CTU size, the PL can work at 220 MHz

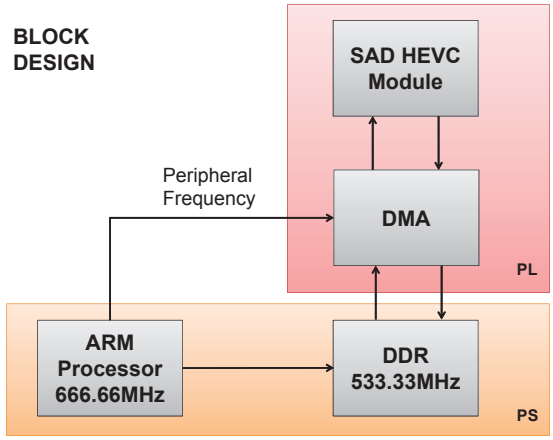


Figure 2.6. Top-level hardware architecture

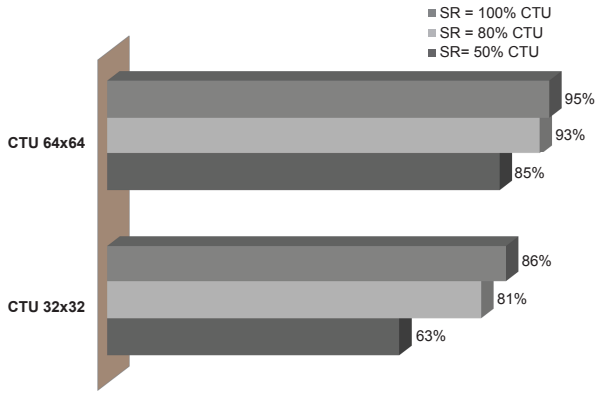
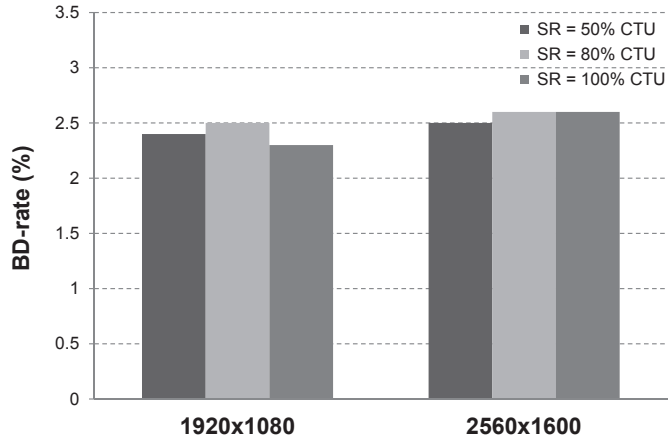


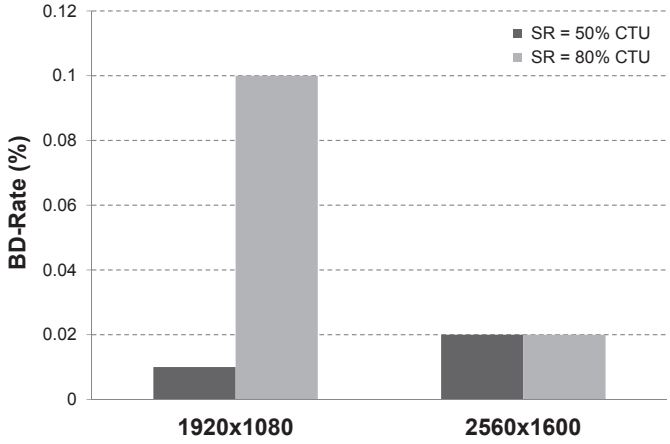
Figure 2.7. Percentage of software reference encoding time required for SAD module with a FS strategy.

whereas with a 32x32 CTU size the PL clock frequency is fixed as maximum in the evaluation board used, 250 MHz.

In order to perform several experiments with the HEVC IME and to observe how the different parameters impact on the Rate Distortion (R/D) performance and coding complexity of the HEVC encoder, the same configurations and models than in the previous section have been choosen (for instance, CTU sizes of 64x64 and 32x32, and their corresponding Search Range (SR) sizes as 100%, 80%, and 50% of the CTU size, and HEVC HM 14 software reference model). In this approach, another two video sequences from the HEVC common conditions video set were selected: ParkScene at 1920x1080 resolution (24 fps) and Traffic 2560x1600 (30 fps).



(a) BD-Rate obtained with a 32x32 CTU size and several SR sizes

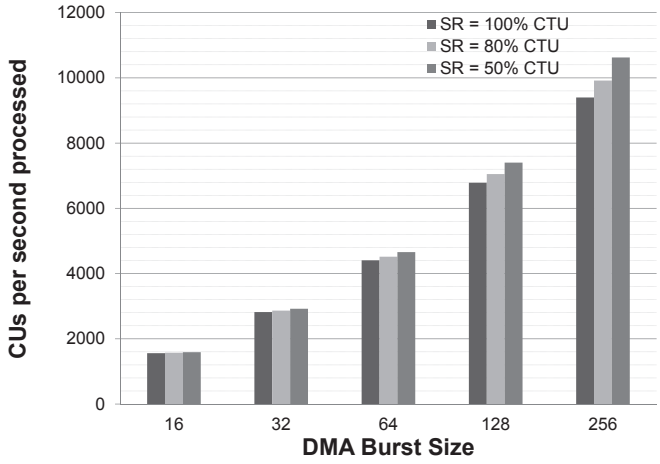


(b) BD-Rate obtained with a 64x64 CTU size and several SR sizes

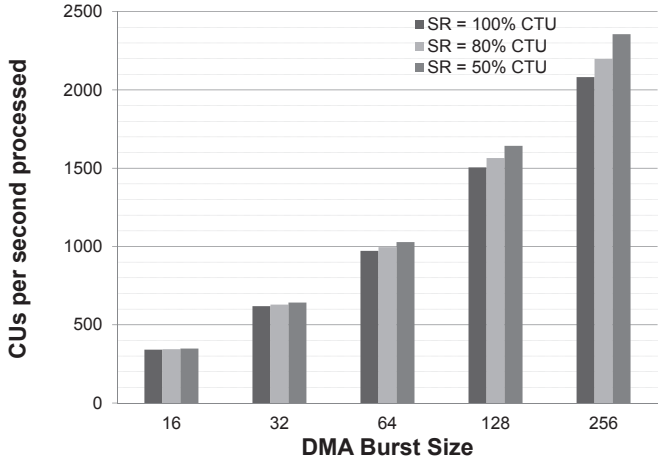
Figure 2.8. BD-Rate values with (a) 32x32, and (b) 64x64 CTU sizes

In Figure 2.7, the percentage of the overall encoding time that was spent by HEVC reference software using the FS algorithm is shown. Generally, the encoder spends more time in the IME module when both the CTU size and Search Range (SR) are higher, as expected. The time spent by HEVC encoder to perform the ME ranges between 63% to 95% of the time required to encode the whole video sequence.

In addition, the impact of the previous parameters on the R/D performance have been analyzed, using the Bjontegaard metric (BD-rate) [24] which allows to compute the average per cent of bitrate overhead/saving between two rate-distortion curves.



(a) CUs per second processed for a 32x32 CTU size



(b) CUs per second processed for a 64x64 CTU size

Figure 2.9. CUs per second processed for each DMA burst size with our proposed SAD HEVC module

In Figure 2.8, it can be observed the BD-rate obtained with different CTU and SR sizes for the two video sequences selected. In order to obtain the percentage of BD-Rate, each R/D curve with its corresponding configuration is compared with a reference R/D curve whose configuration corresponds to a 64x64 CTU size and a search range of ±64 (128x128 search area size). As can be seen, there are slight differences between the SR sizes for a given CTU size, especially for a CTU size of 64x64 where the bitrate penalty is up to 0.1% (see Figure 2.8(b)). In the case of 32x32 CTU size, we obtain higher bitrate overheads, up to 2.7% (see Figure 2.8(a)).

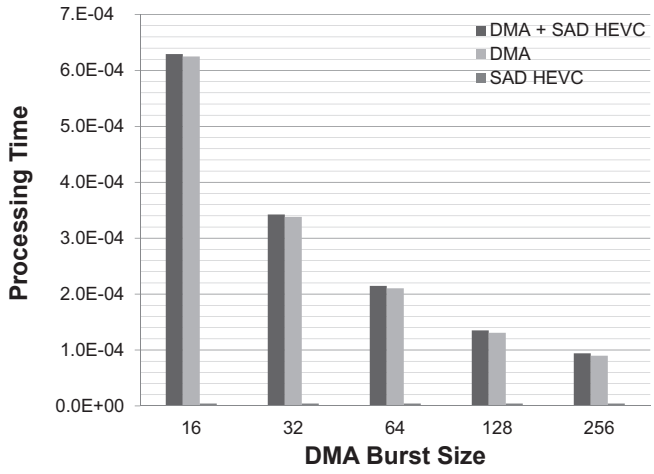


Figure 2.10. Hardware Processing Time for different DMA burst sizes

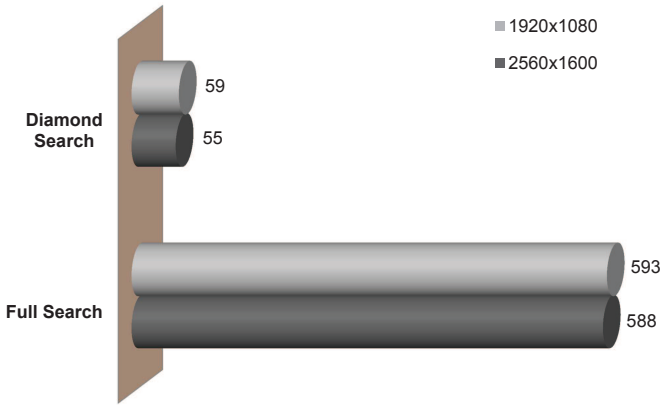


Figure 2.11. Hardware gain (x times) facing software FS and software DS strategies

In Figure 2.9 the number of CUs per second (DMA throughput) transfered for different DMA burst sizes is shown. Note that DMA operations are responsible for transfer data from/to DDR memory and our SAD IME module. In this design, the DMA word size (transfer unit) is set as 32 bits and the burst size can be configured from 16 to 256 words. As can be seen, the throughput increases as the DMA burst size does. That increment is exponential because the time required for the DMA burst initialization is constant and independent on the DMA burst size. Therefore, the maximum burst size of 256, a CTU size of 32x32, and a SR size of 16 is the candidate configuration to achieve the highest DMA transfer throughput in our system.

Taking into account the previous configuration, in Figure 2.10 we show the total DMA time, SAD HEVC module time, and total time to process a CU for different DMA burst sizes. As expected, the maximum DMA burst size provides the best results, requiring the least time to process a CU. Furthermore, depending on the DMA burst size, the differences between DMA transfers and SAD HEVC module processing time vary, being the SAD HEVC module 21x faster with a burst size of 256 and 146x faster with a burst size of 16.

After performing the whole analysis, it can be concluded that the hardware configuration which better adapts to the application requirements (low power consumption, lowest encoding time, reduced video quality loses) is the one that uses a 32x32 CTU size, a SR of 16, and a DMA burst size of 256.

Finally, for a 2560x1600 video resolution, the inclusion of our IME hardware module will speed up the IME computation 55 times and 588 times faster than DS and FS algorithms, respectively, as can be seen in Figure 2.11.

2.2.2 Image coding acceleration

As described in Section 1.3.2, a hardware MPCM codec has been implemented.

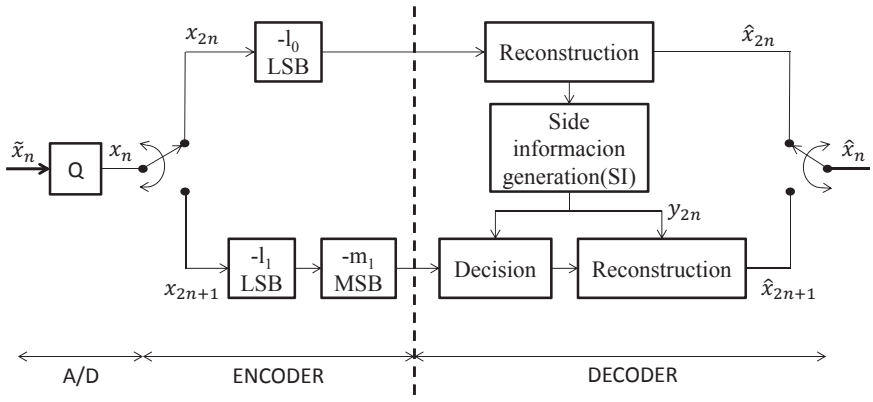


Figure 2.12. Block diagram of MPCM coding algorithm

In Figure 2.12, the MPCM-based coding algorithm is showed. The samples of a continuous-amplitude discrete-time signal ($\widetilde{x_n}$) are divided into sets $S_0 = \{x_{2n} | n \in i = 0, 1, 2, \ldots\}$ and $S_1 = \{x_{2n+1} | n = 0, 1, 2, \ldots\}$, that are encoded with different accuracies. As shown in Figure 2.12, each sample in S_0 is encoded by removing the l_0-LSBs of its codeword (PCM signal) whereas each sample in S_1 is encoded by removing the m_1-MSBs (Most Significant Bits) and l_1-LSBs of its codeword (MPCM signal). The algorithm implemented in this work divides an image into N decimated images, and then it encodes one of the resulting

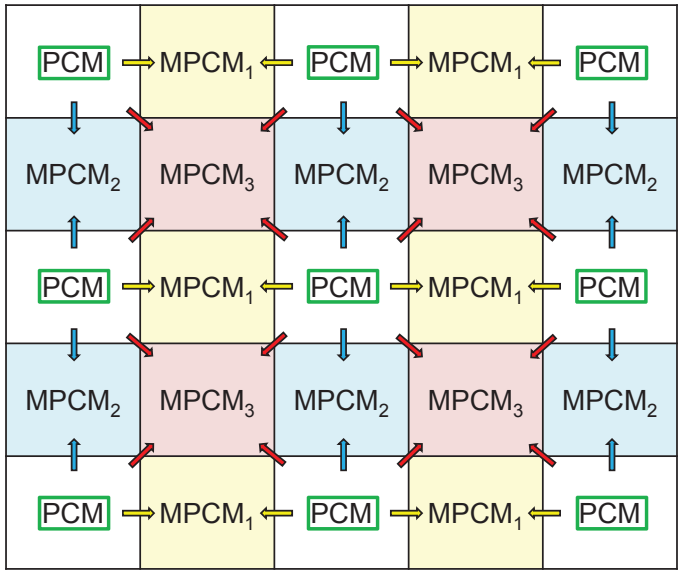


Figure 2.13. Samples in S_0 required to predict samples in S_1 when $N = 4$. $PCM \in S_0$; $MPCM_1, MPCM_2, MPCM_3 \in S_1$

images as sample in S_0 using PCM and encondes the rest of them as samples in S_1 using MPCM.

As the encoding of x_{2n} is equivalent to a PCM signal, the decoder can directly reconstruct the samples of S_0 from their codewords. With respect to the encoded samples in S_1, the decoder follows two steps: decision and reconstruction. First, in order to decide MPCM decoders exploit the correlation between the signal samples by furnishing a prediction for each sample in S_1 based on previously decoded samples in S_0. The Figure 2.13 shows the samples in S_0 required to perform the prediction of each of the samples belonging to S_1, when $N = 4$. The accuracy of the SI depends on the degree of correlation between the samples and the distortion introduced in the encoding of S_0. Furthermore the larger the m_1, the shorter the minimum distance between the codewords of the same set 2^{m_1}, hence, the higher the probability of decision error. As a result, in order to limit these impacts in the encoding algorithm l_0 must be lower than or equal to l_1 ($l_0 \leq l_1$) and m_1 must be the minimum possible.

Once the decoder has estimated the m_1-MSBs, in the reconstruction step, it tries to recover its l_1-LSBs which finally provides a estimated signal $\hat{x}_{2n+1}$. This reconstruction is done by taking the closest value of the chosen interval to the prediction. Notably if the coding parameters accomplish the following characteristics l_0=l_1 and m_1=0, the MPCM encoder/decoder will act as a PCM

Table 2.11. PSNR values for all tested images for a given bit-rate

	PSNR (dB)				
	R=4bpp	R=4.5bpp	R=5bpp	R=5.5bpp	R=6bpp
Imagen	(l_0, l_1, m_1) (1,4,1)	(l_0, l_1, m_1) (2,4,0)	(l_0, l_1, m_1) (3,3,0)	(l_0, l_1, m_1) (1,3,0)	(l_0, l_1, m_1) (2,2,0)
Zelda (512x512)	39.00	38.90	40.15	41.51	45.04
Lena (512x512)	37.74	37.77	39.82	41.06	44.96
Peppers (512x512)	33.70	36.92	39.32	40.29	44.62
Barbara (512x512)	26.06	35.27	38.91	39.83	44.56
Baboon (512x512)	24.45	33.02	37.61	38.20	43.85
Tractor (1920x1080)	38.01	39.67	40.22	42.15	44.73
Woman (2048x2560)	30.53	36.47	39.52	40.70	44.95
Ducks (3840x2160)	35.09	35.00	38.14	38.88	43.72

coding system, since in such cases the help provided by the SI does not compensate the loss suffered by encoding each MPCM signal with fewer bits than the PCM signal. In fact, in these cases, midpoint reconstruction perfoms better than MPCM reconstruction. Nevertheless, in most cases MPCM performs better than or the same as PCM, with great gains at 1, 2, 3 and 4 bpp.

The hardware implementation of both MPCM encoder and decoder was developed over a Zynq-7000 FPGA of Xilinx family, specifically over the ZC702 model which includes the XC7Z020-1CLG484CES SoC (System-on-Chip) [25].

Regarding results, an evaluation of the complete system was performed in terms of PSNR, encoding/decoding times, board area usage, maximum frame rate, and speed-ups comparing results with the one obtained with a CPU sequential algorithm. In these experiments, the results were assessed with eight grayscale images of 8 bits per sample, five of which (Zelda, Lena, Peppers, Barbara, and Baboon) with a resolution of 512x512 pixels, one full-HD (1080p) image, a 2048x2560 image, and finally, a 4K UHD image. Furthermore, the values to the coding/decoding parameters (with $N = 4$, $l_1 = l_2 = l_3$ and $m_1 = m_2 = m_3$) were assigned so as to obtain the best result on average of PSNR for a given bit-rate, although there may be other combinations of parameters that would optimize a particular image [16].

In Table 2.11 the PSNR obtained for all tested images as a function of the bit-rate (R) is presented. As expected, for higher rates (R), which means removing few bits in the encoding process, MPCM algorithm generally

provides good PSNR due to the fact that no significant loss occurs in the coding process, and consequently, not big errors are introduced in the decoding process. Therefore, the lower rate (R), the lower PSNR value.

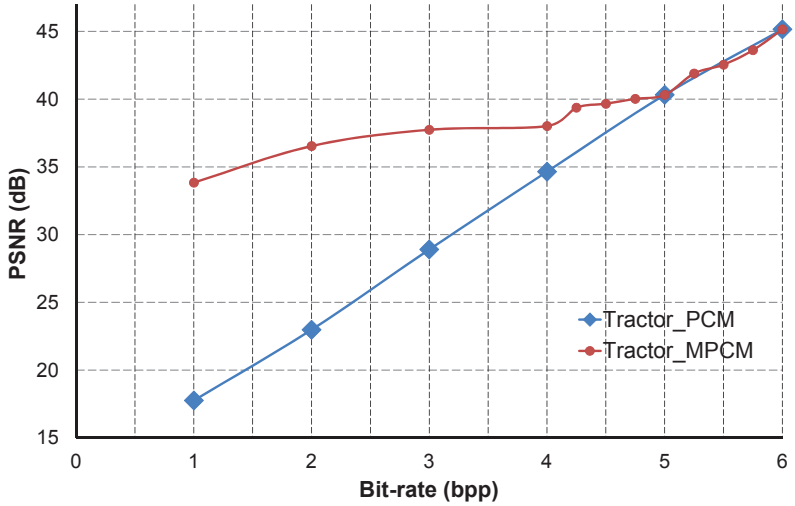


Figure 2.14. PSNR as a function of bitrate using image Tractor encoded with MPCM and PCM.

In Figure 2.14 the comparison between MPCM and PCM coding is shown. PSNR values are represented as function of the rate for the image Full-HD (1080p), Tractor. As can be seen, MPCM obtains the same quality than PCM at low compression rates. However, at high compression rates, MPCM obtains a PSNR improvement up to 15 dB, when compared to PCM. When the previous image is compressed with MPCM, at 6 bpp and 5 bpp non perceptual inequality is observed regarding the original image, as can be seen in Figure 2.15. However, some differences begin to be appreciated at a rate of 4bpp.

In Figure 2.16, the maximum frame rate achievable for the proposed architecture is presented. As shown, the hardware implementation of the MPCM encoder is able to compress up to 3558 frames per second for HD-Ready resolution (720p) or up to 1581 frames per second for Full-HD resolution (1080p). The high speed encoding process makes high speed cameras be able to capture continuously and grab without the restrictions of the internal RAM size. Futhermore, this hardware encoder only uses 1% of all the available area of the FPGA, so it could be used to deploy multiple identical encoders that could run concurrently incrementing the available recording time of a high-speed camera.

(a) Image Full-HD 1920x1080

(b) Decoded Image at 6bpp (44.73dB)

(c) Decoded Image at 5bpp (40.22dB)

(d) Decoded Image at 4bpp (35.44dB)

Figure 2.15. A set of four monochromatic images (Tractor 1920*x*1080) decoded with MPCM: (a) Original image; (b) At 6pp; (c) At 5pp; (d) At 4pp.

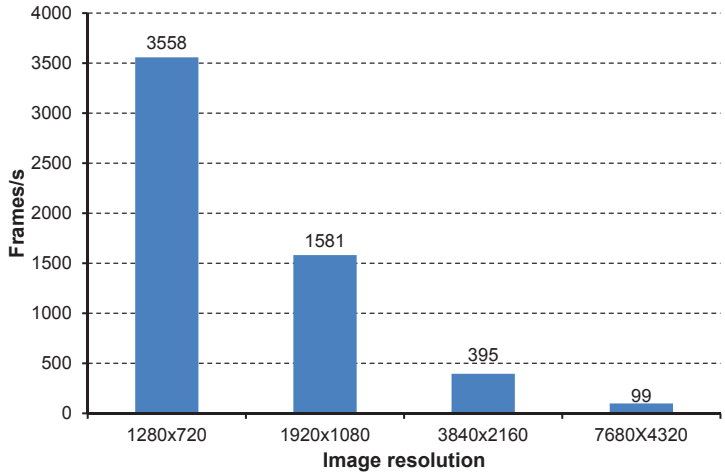


Figure 2.16. Maximum encoded frames per second for different monochromatic image resolutions

As far as the decoder is concerned, in Figure 2.17 the maximum decoding frame rate achievable for the hardware architecture is shown. As can be seen, the hardware implementation of the MPCM decoder is able to recover up to 434 frames per second for HD-Ready (720p) resolution or up to 193 frames per second for Full-HD (1080p) resolution, which corresponds to a throughput of 50 MBytes/s, making available to reproduce high definition cinema at high frame rates. As in encoder, the hardware resources required by the decoder is less than a 1%.

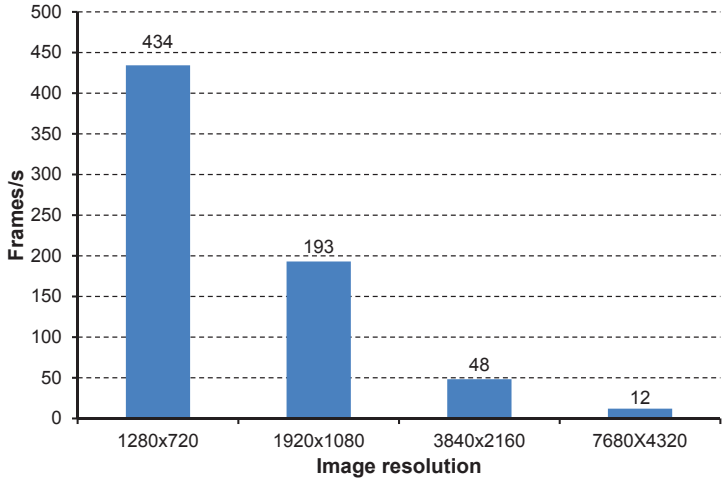


Figure 2.17. Maximum decoder frames per second for different image resolutions

Regarding coding speed, the results show that the MPCM hardware implementation is able to compress a full-HD (1080p) resolution picture at 1581 fps. Indeed, the maximum achievable throughput bandwidth is 409.84 MBytes/s which permits the continuous grabbing of a nowadays high-speed camera at an image resolution of HD-ready and a reasonable good quality. Regarding decoding process, the decoder design is able to recover images at 193 fps for full-HD resolution, with an occupied board area of less than 1%, as in encoder system.

Chapter 3

Conclusions and future work

Contents

3.1 Conclusions and future research lines

In this thesis, the topic of high-resolution video encoding in real-time and even at ultra-high frame rates has been dealt. However, data storage capability, communication bandwidth, processing time and power consumption are critical parameters that should be carefully considered. In order to overcome these limitations, the use of encoders with the best coding efficiency is advisable, such as, in this case, MPCM which has properties similar to those of PCM (low complexity, random access, and scalability) but with a better coding efficiency, or the lastest video coding standard HEVC which improves its predecessor standard H264/AVC by doubling its compression efficiency. Nevertheless, these codecs demand an overwhelming computational cost, especially when removing temporal and spatial redundancy. In order to overcome the previous drawbacks and reduce overall video encoding time, we have worked in the hardware implementation on FPGAs of several codecs' modules. The analysis and assessment of the results described previously in Section 2.2, shows that the proposed solutions obtain a good balance between acceleration of enconding process (complexity reduction) and coding performance.

Firstly, regarding the IME hardware unit for the HEVC video encoder implemented on a Virtex-7 FPGA, this FPGA-based design is able to process video sequences of 2K resolution at 116 fps and 4K video sequences at 30 fps, which represents a huge acceleration of the HEVC video encoding process, achieving real-time encoding for high-definition video contents and beyond. Futhermore, the impact of the configuration parameters, such as maximum CTU and search area sizes, over the encoder complexity and the resulting video quality has been analyzed. The results show that both encoder complexity and encoding time decreases as both the maximum CTU size and the search area do, having a negligible impact in terms of R/D. Differences in R/D are negligible for all configurations, being 2.7% the maximum BD-rate increment for a CTU size of 32x32 and 0.1% for a CTU size of 64x64. In addition, the complete hardware ME architecture including DMA transfers has been evaluated when the previous hardware IME design is applied to a SoC platform Zynq-7 Mini-ITX Motherboard. The results show that throughput increases as the DMA transfer does, being 10,626 the maximum number of CUs per second processed with the fastest configuration (32x32 CTU size, and a SR of 50% of the CTU size). Also, it can be observed that the system bottleneck resides in the DMA transfer process, requiring the 95% of the total processing time of a single CTU. The best DMA burst size in order to obtain the least encoding time is 256 32-bit words. To sum up, with the inclusion of our hardware IME design, the encoding time can be speed-up 558 times when

compared with the software reference HM version.

Secondly, with regard to image coding, the hardware implementation of the MPCM codec is able to compress Full-HD (1080p) resolution images at 1,581 fps. The maximum achievable throughput bandwidth is 409.84 MBytes/s which permits the continuous capturing of a nowadays high-speed camera at an image resolution of HD-ready (1280x720p) and at reasonable good quality. If the final application required a higher image quality, this hardware encoder would be able to give up to 1,640 MBytes/s at a 2:1 compression rate, incrementing the grabbing time over the high-speed camera. Regarding the decoder side, this design is able to recover images at 193 fps of a Full-HD resolution.

Therefore, based on the results obtained and the conclusions of this work, the initial objectives of the thesis have been reached.

Future research lines, which are already underway, will continue in the field of video coding, specifically based on the HEVC standard. In more detail, the following future work is proposed:

- To create a complete application that incorporates our hardware IME architecture as an own IP to the HEVC encoder, using the DMA module and its corresponding drivers, which will be executed on a linux system embedded in the evaluation platform Zynq-7 Mini-ITX Motherboard XC7Z100 (xc7z100ffg900-2).
- To study the different configurations of the DMA module and extend this study to perform an efficient parallel processing that optimizes the whole IME design, trying to eliminate the bottleneck generated by the DMA.
- To include several optimizations to the motion estimator, such as adding the R/D computation.
- To study the next most computationally complex modules of the HEVC encoder, such as Intra prediction (removal of spatial redundancy), and create a new hardware module that implements these functions.

3.2 Conclusiones y futuras líneas de investigación

En esta tesis, se ha tratado el tema de la codificación de vídeo de alta resolución en tiempo real e incluso a velocidades de frame ultra-altas. Sin embargo, la capacidad de almacenamiento de datos, el ancho de banda de comunicación, el tiempo de procesamiento y el consumo de energía son parámetros críticos que deben ser cuidadosamente considerados. Con el fin de

superar estas limitaciones, es aconsejable el uso de codificadores con la mejor eficiencia de codificación posible, tal como, en este caso, el códec MPCM y el estándar HEVC. El códec MPCM tiene propiedades similares a las del codificador PCM (baja complejidad, acceso aleatorio y escalabilidad) pero con una mejor eficiencia de codificación, y el último estándar de codificación de vídeo HEVC que mejora al estándar previo, H264/AVC, duplicando su eficiencia de compresión. Sin embargo, estos códecs tienen una complejidad y un coste computacional abrumador, especialmente en los bloques responsables de eliminar la redundancia temporal y espacial. Con el fin de superar los inconvenientes anteriores y reducir el tiempo total de codificación de vídeo, en la presente investigación se ha trabajado en la implementación hardware sobre FPGAs de varios módulos de estos códecs. El análisis y la evaluación de los resultados obtenidos y descritos anteriormente en la Sección 2.2 muestran que las soluciones propuestas son muy ventajosas, obteniendo una considerable aceleración en la codificación de imagen y vídeo.

En primer lugar, se ha presentado una unidad IME hardware del codificador de vídeo HEVC implementado en un FPGA Virtex-7. Este diseño basado en FPGA es capaz de procesar formatos de vídeo 2K a 116 fps y secuencias de vídeo 4K a 30 fps, lo que representa una enorme aceleración del proceso de codificación de vídeo con HEVC, logrando codificar en tiempo real contenidos de vídeo de alta definición y más. Además, se ha analizado el impacto de los parámetros de configuración sobre la complejidad del codificador y la calidad de vídeo resultante, tales como los tamaños del CTU máximo y del área de búsqueda. Los resultados muestran que tanto la complejidad del codificador como el tiempo de codificación disminuyen según lo hacen los tamaños del CTU y del área de búsqueda, teniendo un mínimo impacto en términos de R/D. Las diferencias en R/D son insignificantes para todas las configuraciones, siendo el incremento máximo de BD-rate un 2,7% para un tamaño de CTU de 32x32 y un 0,1% para un tamaño de CTU de 64x64. Además, también se ha evaluado la arquitectura hardware del IME completo, incluyendo las transferencias DMA, cuando el diseño IME hardware se implementa sobre una placa de evaluación Zynq-7 Mini-ITX SoC. Los resultados muestran que el rendimiento aumenta a medida que el tamaño de la ráfaga de transferencia lo hace, siendo 10.626 los CUs por segundo máximos procesados, con la configuración más rápida (CTU de 32x32 píxeles, y un SR del 50% del tamaño del CTU). Asimismo, se puede observar que el cuello de botella del sistema reside en el proceso de transferencia del DMA, requiriendo el 95% del tiempo total de procesamiento de un CTU. El mejor tamaño de ráfaga del DMA para obtener el menor tiempo de codificación es de 256 palabras de 32 bits. En resumen, con la inclusión de nuestro diseño hardware del IME, el tiempo de codificación

puede acelerarse 558 veces en comparación con la versión software del IME.

En segundo lugar, con respecto a la codificación de imágenes, la implementación hardware del códec MPCM es capaz de comprimir una imagen de resolución full-HD (1080p) a 1.581 fps. El ancho de banda máximo que se alcanza es de 409,84 MBytes/s, lo que permite la captura continua de una cámara de alta velocidad con una resolución de imagen de 1280x720p y una calidad razonable. Si la aplicación final requiriese una calidad de imagen superior, este codificador hardware podría alcanzar hasta 1.640 MBytes/s a una tasa de compresión de 2:1, incrementando el tiempo posible de grabación en la cámara de alta velocidad. Con respecto al bloque del decodificador, este diseño es capaz de recuperar imágenes a 193 fps a una resolución full-HD.

Por tanto, en base a los resultados obtenidos y las conclusiones de este trabajo, podemos asegurar que se han alcanzado los objetivos iniciales de la presente tesis.

Las líneas de investigación futuras, que ya están en marcha, continuarán en el campo de la codificación de vídeo, en concreto, basándose en el estándar HEVC. En concreto, se proponen los siguientes trabajos futuros:

- Crear una aplicación completa que incorpore nuestra arquitectura hardware IME como un IP intrínseco al codificador HEVC, mediante el módulo DMA y sus drivers correspondientes, que será ejecutado sobre un sistema linux embebido en la plataforma de evaluación Zynq-7 Mini-ITX Motherboard XC7Z100 (xc7z100ffg900-2).
- Estudiar las diferentes configuraciones del módulo DMA y extender este estudio para realizar un procesamiento en paralelo más eficiente que optimice el diseño completo del IME, intentando eliminar el cuello de botella que el DMA genera.
- Incluir diversas optimizaciones al estimador de movimiento, como por ejemplo, añadir el cálculo del R/D.
- Estudiar el resto de los módulos más costosos computacionalmente del codificador HEVC, como puede ser la predicción Intra (eliminación de la redundancia espacial), y crear nuevos módulos hardware que implementen dichas funciones.

3.3 Other publications

The author of the present thesis has participated in the following additional publications, related to the implementation of image/video codecs on FPGAs:

- Estefania Alcocer, Otoniel Lopez-Granado, Roberto Gutierrez, and Manuel P. Malumbres, “Evaluation of HEVC hardware IME using SoC platform”, XXXI Edition of Design of Circuits and Integrated Systems Conference (DCIS2016), Granada, Noviembre 2016.
- Estefania Alcocer, Otoniel Lopez-Granado, Roberto Gutierrez, and Manuel P. Malumbres, “Implementation of a low latency motion estimator for HEVC encoder on FPGA”, 3rd International Conference on Advances in Information Processing and Communication Technology (IPCT15), the IRED, Roma, December 2015.

Bibliography

[1] Youn-Long Steve Lin, Chao-Yang Kao, Hung-Chih Kuo, and Jian-Wen Chen, editors. *VLSI Design for Video Coding*. Springer, 2010.

[2] Mpeg the moving picture experts group. http://mpeg.chiariglione.org/. Accessed 28 April 2017.

[3] Itu-t standardization on visual coding - the video coding experts group vceg. http://www.itu.int/en/ITU-T/studygroups/com16/video/Pages/default.aspx. Accessed 28 April 2017.

[4] S. M. Trimberger and J. J. Moore. Fpga security: Motivations, features, and applications. *Proceedings of the IEEE*, 102(8):1248–1265, Aug 2014.

[5] Dong-U Lee. *Hardware Designs for Function Evaluation and LDPC Coding*. PhD thesis, Imperial College, London, 2004.

[6] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, Dec 2012.

[7] Xilinx 7 Series FPGAs. Virtex-7 t and xt fpgas data sheet. https://www.xilinx.com/support/documentation/data_sheets/ds183_Virtex_7_Data_Sheet.pdf. Accessed 2 May 2017.

[8] Hevc software repository hm–14.0 reference model. https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-14.0. Accessed 26 April 2017.

[9] Vision Research and Ametek Materials Analysis Division. Phantom ultrahigh-speed cameras. http://http://www.phantomhighspeed.com/products/ultrahigh-speed-cameras. Accessed 28 March 2017.

[10] Photron. Fastcam sa series. https://photron.com/product-category/cameras/sa-series/. Accessed 28 March 2017.

[11] Fastec Imaging. Hispec 5. http://www.fastecimaging.com/products/tethered-cameras/hispec-5. Accessed 28 March 2017.

[12] ix Cameras. i-speed 3 series. http://www.ix-cameras.com/3-Series/. Accessed 28 March 2017.

[13] Nuggehally S Jayant and Peter Noll, editors. *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, Englewood Cliffs, 1984.

[14] Peter Gemeiner, Wolfgang Ponweiser, Peter Einramhof, and Markus Vincze. Real-time slam with a high-speed cmos camera. In *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, pages 297–302, Sept 2007.

[15] J. Prades-Nebot, A. Roca, and E. Delp. Modulo-pcm based encoding for high speed video cameras. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 153–156, oct. 2008.

[16] Marleen Morbee. *Optimized information processing in resource-constrained vision systems*. PhD thesis, Universidad PolitÃ©cnica de Valencia, Universiteit Gent, 2011.

[17] Xilinx. Vivado design suite user guide. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug910-vivado-getting-started.pdf. Accessed 9 May 2017.

[18] Frank Bossen. Common test conditions and software reference configurations. Technical report, Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

[19] Xilinx. Xilinx zynq-7000 all programmable soc zc702 evaluation kit. https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html. Accessed 9 May 2017.

[20] Xilinx. Avnet zynq-7000 all programmable soc mini-itx development. https://www.xilinx.com/products/boards-and-kits/1-4b47l9.html. Accessed 9 May 2017.

[21] A. Medhat, A. Shalaby, M. S. Sayed, M. Elsabrouty, and F. Mehdipour. A highly parallel sad architecture for motion estimation in hevc encoder. In *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 280–283, Nov 2014.

[22] T. D'huys, S. Momcilovic, F. Pratas, and L. Sousa. Reconfigurable data flow engine for hevc motion estimation. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1223–1227, Oct 2014.

[23] Xu Yuan, Liu Jinsong, Gong Liwei, Zhang Zhi, and R. K. F. Teng. A high performance vlsi architecture for integer motion estimation in hevc. In *2013 IEEE 10th International Conference on ASIC*, pages 1–4, Oct 2013.

[24] G. Bjontegaard. Document vceg-m33: Calculation of average psnr differences between rd-curves. Technical report, ITU-T VCEG Meeting, Austin, Texas, USA, Tech. Rep, 2001.

[25] Xilinx Zynq-7000. Zynq-7000 all programmable soc overview, advance product specification - ds190 (v1.2). http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf. Accessed 26 April 2017.

Appendix I

Acronyms

AMP	Asymmetric Motion Partition
AP	All Programmable
AV	Audio and Visual
AVC	Advanced Video Coding
BRAM	Block Random Access Memory
CLB	Configurable Logic Block
CTU	Coding Tree Unit
CU	Coding Unit
DDR	Double Data Rate
DMA	Direct Memory Access
DS	Diamond Search
fps	frames per second
FPGA	Field Programmable Gate Array
FS	Full Search
Gb	Gigabit
HD	High Definition
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
IME	Integer Motion Estimation
IP	Intellectual Property
I/O	Input-Output
LUT	Look-up table
ME	Motion Estimation
MPCM	Module Pulse Code Modulation
MV	Motion Vector
PCM	Pulse Code Modulation

PE	Processing Element
PL	Programmable Logic
PS	Processing System
PSNR	Peak Signal-to-Noise Ratio
PU	Processing Unit
R/D	Rate Distortion
RAM	Random Access Memory
ROM	Read Only Memory
RTL	Register-Transfer Level
SAD	Sum of Absolute Differences
SATB	SAD Adder Tree Block
SI	Side Information
SDRAM	Synchronous Dynamic Random Access Memory
SoC	System-On-Chip
SPS	Sequence Parameter Set
SR	Search Range
UHD	Ultra High Definition
VRML	Virtual Reality Modeling Language

Appendix II

Articles

This dissertation has been carried out in the modality of a doctoral thesis presented with a set of publications. According to the Internal Regulations of the Miguel Hernandez University for the presentation of doctoral theses with a set of publications, the articles presented in their original language are attached in this appendix.

- **Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder**
  Estefania Alcocer, Roberto Gutierrez, Otoniel Lopez-Granado, and Manuel P. Malumbres
  Journal of Real-Time Image Processing. Springer Verlag Berlin Heidelberg 2016
  ISSN 1861-8219
  http://dx.doi.org/10.1007/s11554-016-0572-4

- **MPCM: a hardware coder for super slow motion video sequences**
  Estefania Alcocer, Otoniel Lopez-Granado, Roberto Gutierrez, and Manuel P. Malumbres
  EURASIP Journal on Advances in Signal Processing 2013. Springer Open Journal
  ISSN 1687-6180
  http://dx.doi.org/10.1186/1687-6180-2013-142

Next, the classification of the journals where the articles are published within the Journal Citation Reports index.

- **Journal of Real-Time Image Processing**
  Impact factor: 1.564
  Category Name: ENGINEERING, ELECTRICAL & ELECTRONIC
  Total Journals in Category: 257
  Journal Rank in Category: 104
  Quartile in Category: **Q2**
  Year: 2015

- **EURASIP Journal on Advances in Signal Processing**
  Impact factor: 0.808
  Category Name: ENGINEERING, ELECTRICAL & ELECTRONIC
  Total Journals in Category: 248
  Journal Rank in Category: 164
  Quartile in Category: **Q3**
  Year: 2013

The following list indicates the affiliation of the articles' coauthors presented in this thesis.

- **Estefanía Fátima Alcocer Espinosa**
  ealcocer@umh.es
  Materials Science, Optics and Electronic Technology department
  Miguel Hernández University
  03202, Elche, Alicante, Spain

- **Roberto Gutiérrez Mazón**
  roberto.gutierrez@umh.es
  Communication Engineering department
  Miguel Hernández University
  03202, Elche, Alicante, Spain

- **Otoniel Mario López Granado**
  otoniel@umh.es
  Physics and Computer Architecture department
  Miguel Hernández University
  03202, Elche, Alicante, Spain

- **Manuel José Pérez Malumbres**
  mels@umh.es
  Physics and Computer Architecture department
  Miguel Hernández University
  03202, Elche, Alicante, Spain

J Real-Time Image Proc
DOI 10.1007/s11554-016-0572-4

ORIGINAL RESEARCH PAPER

Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder

Estefania Alcocer[1] · Roberto Gutierrez[2] · Otoniel Lopez-Granado[1] · Manuel P. Malumbres[1]

Received: 30 September 2015 / Accepted: 26 February 2016

Abstract High-Efficiency Video Coding (HEVC) was developed to improve its predecessor standard, H264/AVC, by doubling its compression efficiency. As in previous standards, Motion Estimation (ME) is one of the encoder critical blocks to achieve significant compression gains. However, it demands an overwhelming complexity cost to accurately remove video temporal redundancy, especially when encoding very high-resolution video sequences. To reduce the overall video encoding time, we propose the implementation of the HEVC ME block in hardware. The proposed architecture is based on (a) a new memory scan order, and (b) a new adder tree structure, which supports asymmetric partitioning modes in a fast and efficient way. The proposed system has been designed in VHDL (VHSIC Hardware Description Language), synthesized and implemented by means of the Xilinx FPGA, Virtex-7 XC7VX550T-3FFG1158. Our design achieves encoding frame rates up to 116 and 30 fps at 2 and 4K video formats, respectively.

Keywords HEVC · FPGA · Integer motion estimation · Inter-prediction · SAD architecture

✉ Estefania Alcocer
ealcocer@umh.es

[1] Physics and Computer Architecture department, Miguel Hernandez University of Elche, Alicante, Spain

[2] Communication Engineering department, Miguel Hernandez University of Elche, Alicante, Spain

1 Introduction

The High-Efficiency Video Coding (HEVC) standard is the most recent joint video project of the ITU-T VCEG and ISO/IEC MPEG standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [1]. Previous video coding standards are currently used for many applications such as broadcast of High-Definition (HD) TV, video content acquisition, Internet and mobile video streaming, and real-time conversational applications. However, new video services with UltraHigh-Definition (UHD) formats are emerging, which need higher coding efficiency than previous standards. HEVC has been designed to deal with these demands, working with higher video resolutions and adapting its design to allow the use of parallel processing techniques. It can compress video about twice as much as its predecessor, H264/AVC, without sacrificing quality, providing video delivery with higher resolutions and frame rates, higher dynamic range, and a wider color gamut. Furthermore, HEVC contains new key features that are friendly with the use of parallel processing techniques [2].

As in previous video standards, Motion Estimation (ME) is one of the most critical modules in the video encoding process since it is able to efficiently remove the temporal redundancy between successive frames. However, the ME module is by far the most complex task of the encoder, requiring more than 90 % of the encoding time [3].

In HEVC, the complexity is even more critical due to several issues such as (a) a large set of Coding Tree Unit (CTU) partitioning modes, (b) the presence of multiple reference frames, and (c) the varying size of Coding Units (CU) in comparison with its predecessor H264/AVC. In addition, HEVC adopts Variable Block Size Motion Estimation (VBSME) to obtain advanced coding efficiency,

Published online: 18 March 2016

which comes at the expense of a huge increase of computational complexity.

For these reasons, several hardware architectures have been proposed to speed up the HEVC ME module, reducing the overall encoder complexity as much as possible. The Integer-pel Motion Estimation (IME) block is in charge of motion estimation and it is composed of (a) an integer motion search algorithm, and (b) a Rate/Distortion (R/D) optimization procedure that optimally reduces the temporal redundancy found at the video sequence. In most of the works found in the literature, the proposed IME hardware architectures are only focused on the motion search algorithm since it takes most of the computational complexity of the IME block. There are a lot of motion search algorithms that can be used to find the motion in video sequences. The most popular in hardware implementations is the Full Search (FS) algorithm. It follows greedy behavior by searching for motion at all points of the established search area of a reference frame, and, as a consequence, it is able to provide an optimal result (i.e., a motion vector that minimizes the residual error of the actual CTU).

The architecture proposals in [3, 4, 6, 7, 9] present an IME hardware block using FS strategy. In [3], a Sum of Absolute Differences (SAD) unit on a Field-Programmable Gate Array (FPGA) is proposed that is able to test all partition modes of a CTU except the set of asymmetric partition modes. Authors fixed a search area size lower than the one established by the standard, being able to run as fast as 30 fps at 2k video resolutions. The work presented in [4] proposes a SAD unit that computes all CTU partitions, achieving the same frame rates as previous work at 4k video formats. In their proposal, the search area has the same size as the maximum CTU, being implemented on an Application-Specific Integrated Circuit (ASIC). In [6], the maximum CTU size is reduced to 32×32 with a search area size of ± 23 pixels. This architecture is implemented on an FPGA device and achieves 30 fps at 1080p video resolutions. Different configurable search areas are studied in [7], achieving a maximum frame rate of 57 fps for a 720p video resolution. Several SAD units implemented on FPGA are described in [9], with different levels of parallelization, but no data about search area size, memory management, or how they obtain the minimum SAD are included.

On the other hand, [5, 8, 15] have proposed architectures which increase the throughput by limiting the number of searches in the reference frame. In [15], a motion estimation system for the HEVC encoder is presented. This design includes both integer-pel and fractional-pel motion estimation, achieving video encoding speeds of 1080p@60fps and 2160p@30fps when implemented over FPGA and ASIC technologies, respectively. The process in [15] is interrupted when the number of motion searches arrives at a limit fixed for a given resolution.

In addition, in [5] and [8], different implementations of suboptimal motion search strategies called fast ME algorithms, such as new Diamond Search (DS) or new Three Step Search (TSS), are shown. Similar hardware ME architectures have also been studied for the previous H264/AVC standard in [10–14], which are of interest for our work due to the high similarity of the IME block architecture in both standards.

Therefore, our purpose is to design a new hardware architecture that may perform IME computation in a fast and accurate way to significantly reduce the computation cost of the overall encoder. We will use FPGA technology, since it encourages design reuse and can greatly enhance the upgradability of digital systems. The programmability of FPGAs is particularly useful for highly flexible encoding systems that can accommodate a multitude of existing standards as well as the emergence of new ones [12].

Regarding the novelty of the proposed architecture, we present both innovative techniques: (a) a new SAD adder tree structure, and (b) a new memory scan order.

Firstly, we designed a new SAD adder tree structure to perform the additions at the first level of the tree, starting from the maximum size of the CTU, and halving the amount of additions at the next tree levels. This approach is different from the rest of state-of-the-art works, which divide a CTU into smaller blocks for consecutive accumulations, keeping the same additions in each step and thus requiring a higher number of steps to acquire all SADs. With our proposal, we took advantage of the resources provided by the FPGA, obtaining the minimum possible latency when calculating SADs of all levels and partitions for a CTU. In this way, SADs corresponding to asymmetric partitions are obtained in a fast and efficient way.

Secondly, regarding the new memory scan order, a series of reconfigurable shift registers and processing elements are responsible for storing the necessary pixels of both reference and current frames, keeping them always available for calculating the SADs and MVs of a CTU. With our system, we avoid external memory accesses since available data are highly reused by reconfiguring the displacement in a more efficient way.

The rest of the paper is organized as follows. Section 2 describes the HEVC ME module. Section 3 presents the architecture design of the proposed ME system while in Sect. 4, implementation results are provided in terms of hardware resources, time encoding, and R/D performance. Finally, in Sect. 5 some conclusions are drawn.

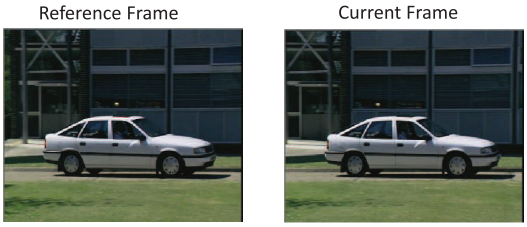
2 HEVC motion estimation

The motion estimation technique is based on the similarity between adjacent video frames, predicting the current frame based on a previous or subsequent reference frame in order of appearance.

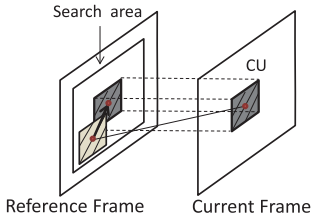
The Motion Vector (MV) represents the translational movement of a picture area in the current frame compared to its position in the reference frame. This movement is found inside a defined search area to bound the overall motion search complexity, as shown in Fig. 1.

In the ME process, each video frame is subdivided and partitioned into basic coding units called CTUs. The coding structure in HEVC consists of CUs with a maximum size of 64 × 64 pixels, as large as that of CTUs, which can be recursively divided into picture squares until achieving a block size of 8 × 8 pixels. Each CU consists of prediction units (Intra- or Inter-) and its size can vary from the maximum size of the CU to 4 × 4 pixels for Intra prediction, or to 4 × 8 or 8 × 4 for inter-prediction, supporting 8 partitioning modes as shown in Fig. 2. Prediction units of sizes 2N × 2N and N × N are called square motion partitions (square); 2N × N and N × 2N as Symmetric Motion Partitions (SMP); and 2N × nU, 2N × nD, nL × 2N, and nR × 2N as Asymmetric Motion Partitions (AMP). The total number of different partitions for a 64 × 64 CTU is more than 600, and for each of these partitions, the HEVC encoder performs one ME process to determine the best CTU partitions in terms of bit rate and video quality.

There are many kinds of algorithms for block-based IME. The most accurate strategy is the FS algorithm, which exhaustively finds motion for all prediction unit blocks at every single point of the established search area. Due to computational regularity and excellent video quality, FS motion estimation is commonly employed in hardware implementations [16]. Therefore, we will focus our work towards the design and hardware implementation of an FS algorithm that is able to significantly speed up the motion estimation process of the HEVC encoder without losing R/D performance.



(a) Current and reference frames



(b) Obtaining motion vector

Fig. 1 Motion estimation

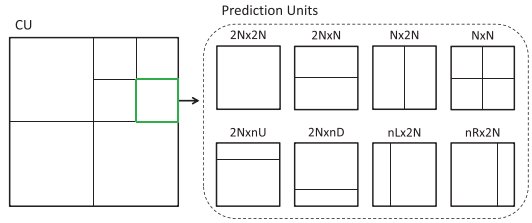


Fig. 2 Predictions units within a CU

3 Hardware architecture

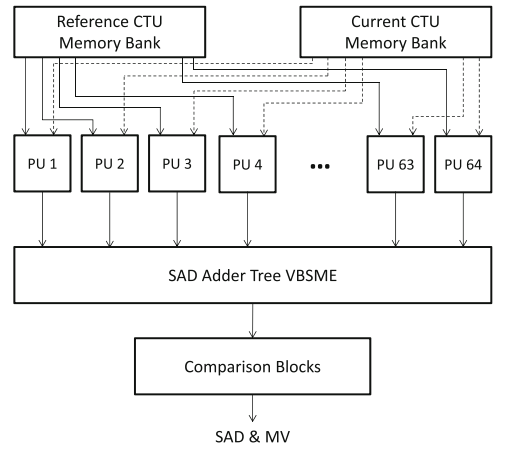
In this section, we present a high-performance IME hardware unit in HEVC that provides the minimum SADs and associated MVs of all possible partitions from a 64 × 64 CTU for inter-prediction, exploiting parallelism in an efficient way. The system is composed of memory areas for current CU and reference search area pixels, 64 Processing Units (PU), one SAD Adder Tree Block (SATB), and one comparison block that saves the minimum SAD values and their corresponding MVs for all CU partitions. Figure 3a shows the proposed hardware architecture.

As shown in Fig. 3b, one PU consists of 64 Processing Elements (PEs), where each PE computes the difference of both the current and the reference pixel (see Fig. 3c). So each PU calculates the distortion values of a column of 64 pixels. At each clock cycle, current and reference pixel columns are delivered to the 64 PUs, being able to compute the pixel distortion values of a 64 × 64 block (maximum CU size) just in one clock cycle, that is, all distortions needed to obtain the SAD of a 64 × 64 CU in a particular position of the search area are calculated in just one clock cycle. The next block in our system, SATB, computes the SADs for all the possible prediction units (more than 600) by properly grouping the 64 × 64 pixel distortions obtained before.

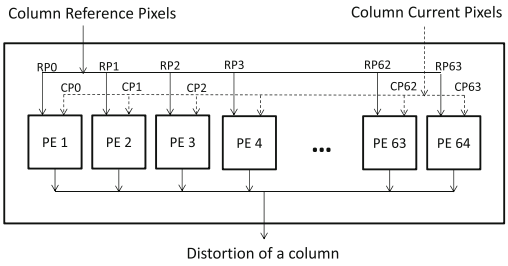
The process described above is performed for each of the positions of the search area, delivering the SADs to the comparison block, which is in charge of storing the minimum SADs with their corresponding MVs for each prediction unit of current CU. Table 1 lists the total number of different SAD partitions for a 64 × 64 CU.

3.1 Memory read controller block

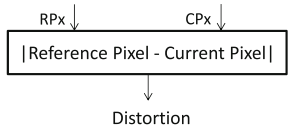
The memory read controller block is composed of a Block-RAM (BRAM) memory and a set of shift registers.



(a) Hardware block architecture



(b) Processing Unit (PU)



(c) Processing Element (PE)

Fig. 3 Proposed IME architecture

A BRAM consists on an embedded memory block within the FPGA. Pixels belonging to the search area of the reference frame are stored in the BRAM and current CTU pixels are saved in each PE. The reference pixels are spread from BRAM to the set of shift registers that are responsible for feeding PEs to calculate the distortion of the current CTU in a particular search area position.

The search area is just centered on the location of current CTU and the default search window spans ± 64 pixels from the current CTU position, which defines a 128×128 search area as shown in Fig. 4, that is, the current CTU will be matched in 128×128 different pixel positions, being necessary to load on BRAM memory the pixels belonging to a reference frame area of 191×191 pixels.

To provide high data reuse, a snake scan order and a reconfigurable data path with 64 propagation registers are adopted. The snake scan order visits all positions of the search area following a Hamiltonian path composed by consecutive vertical scans with alternating directions (the first vertical scan begins from top to bottom, then moves one pixel to the right and starts the next vertical scan in a bottom to up direction, and so on) as illustrated in Fig. 4. So, there are three scanning directions U (upward), D (downward), and R (rightward).

The current 64×64 CTU pixels are stored in the PEs only once (at the beginning). The reference pixels will also be loaded to the PEs but instead of loading from BRAM will be loaded from the shift registers, since they will help us to perform the snake scan order and as a consequence a huge reduction of memory load operations will be achieved.

So, the memory controller will be the one that manages the shift registers set by loading rows of 64 pixels from the reference frame area (BRAM) and performing the shift register operations to cope with the snake scan order.

In Fig. 5, a diagram with the shift register set and the loading and shifting operations is shown. At the beginning, the register set is empty, so we have to perform several (64) load and shift operations before calculating the first SAD. As can be seen in Fig. 5, the first 64 clock cycles are dedicated to load the first 64-pixel rows starting from the left most upper position of the search area, following a downward (D) scan direction. In this figure, each 64-pixel row is labeled with the (x,y) pixel locations of the reference frame area. After loading the 64x64 reference frame block, all the pixels are sent to the PEs to compute the SAD in just one cycle (remember that the actual 64×64 CU pixels are already stored in the PEs waiting for this operation). At this point, the first SAD is computed. After that, we proceed in the D scan direction to compute the SAD of the next search area position. For this purpose, we only need to load an additional 64-pixel row in the D scan direction. So, in one clock cycle, (a) a right-shift operation takes place, discarding the first pixel row stored in the shift register 63, and (b) the new pixel row is loaded from BRAM in shift register 0. Then, the 64×64 pixels stored in the shift registers are sent to the PEs to compute a new SAD.

After computing the last SAD in the downward scanning direction, we have to change the scan direction from D to R, following the snake pattern described before. Moving the search area position one pixel to the right could be easy if we simply shift to the left one pixel in all shift registers (see Fig. 5 at the R scan direction). So, shift registers will

Table 1 Total number of SADs for each partition in a 64 × 64 CU

Block size	No. of SADs	Block size	No. of SADs
64 × 64 (2N × 2N)	1	32 × 32 (2N × nU)	8
64 × 64 (2N × N)	2	32 × 32 (2N × nD)	8
64 × 64 (N × 2N)	2	16 × 16 (2N × 2N)	16
64 × 64 (N × N)	4	16 × 16 (2N × N)	32
64 × 64 (nL × 2N)	2	16 × 16 (N × 2N)	32
64 × 64 (nR × 2N)	2	16 × 16 (N × N)	64
64 × 64 (2N × nU)	2	16 × 16 (nL × 2N)	32
64 × 64 (2N × nD)	2	16 × 16 (nR × 2N)	32
32 × 32 (2N × 2N)	4	16 × 16 (2N × nU)	32
32 × 32 (2N × N)	8	16 × 16 (2N × nD)	32
32 × 32 (N × 2N)	8	8 × 8 (2N × 2N)	64
32 × 32 (N × N)	16	8 × 8 (2N × N)	128
32 × 32 (nL × 2N)	8	8 × 8 (N ×2 N)	128
32 × 32 (nR × 2N)	8	Total	677

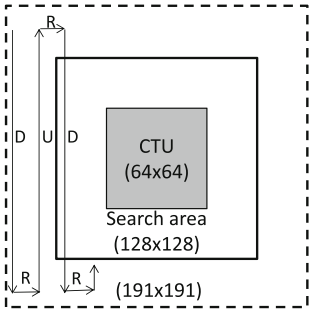


Fig. 4 Scan order of the search area

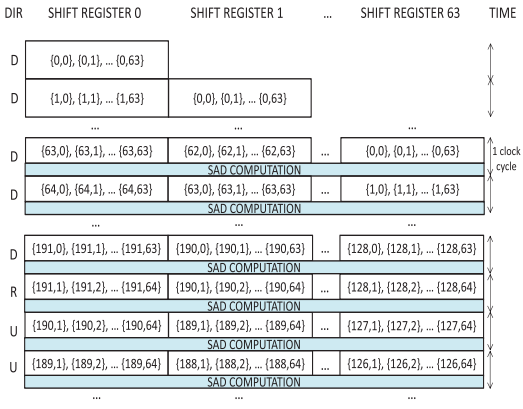


Fig. 5 Shift registers set: loading and shifting operations

contain the 64 × 64 search area block corresponding to the new position, and ready for the corresponding SAD computation.

After computing this SAD, we again change the scan direction from R to U, so we need to load a new 64-pixel row from BRAM, but now the loading is performed in the last shift register (63) and the register shift operation will be set to the left, discarding the contents of the first shift register (0).

The new SAD may now be computed, and as the scan direction is upwards, loading and shifting operations will be performed in the same way until a new change in scan direction is found.

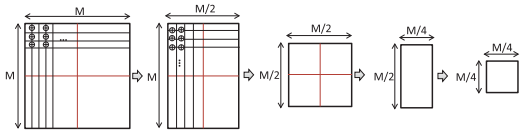
This procedure will iterate until all searching area positions have been processed, providing one SAD at every clock cycle to the next module in the proposed architecture, the SATB.

3.2 SAD adder tree block

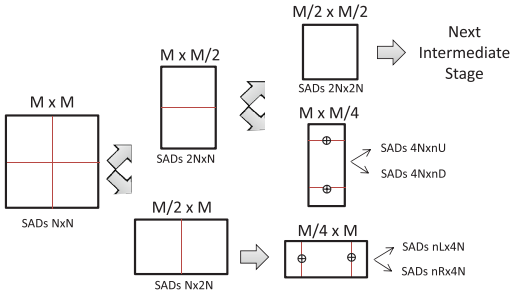
The SATB block is in charge of computing the SAD values for all partitions of each 64 × 64 CTU at every clock cycle. For inter-prediction, the HEVC standard proposes a partition size that ranges from 64 × 64 (maximum CU size) to 4 × 8/8 × 4 with different shapes—square, symmetric, and asymmetric partitions. After receiving the 64 × 64 distortions associated to the current search area position, a succession of aggregation stages are performed in this block to compute the corresponding SAD values for all the CTU partitions (a total number of 677), as shown in Fig. 6.

At the first stage, Fig. 6a, all pairs of consecutive distortion columns/rows of the input 64 × 64 SAD block (M = 64) are added, reducing the width/height of the resulting partition by one-half, until the block size of these added distortions is reduced to 16 × 16, from which the first SADs are obtained.

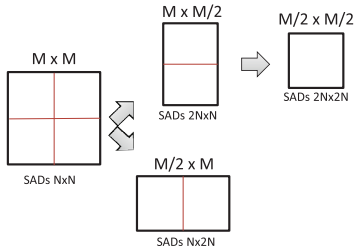
At the next three intermediate stages, a similar process to the one described above is followed. The successive



(a) First stage: M=64



(b) Three intermediate stages: M=16 and N=4; M=8 and N=8; M=4 and N=16



(c) Last stage: M=2 and N=32

Fig. 6 Structure of the SAD adder tree block

sums of different configurations (row–column, column–column, row–row) are performed to get the SADs of all partitions of a 64 × 64 CTU. For instance, in the first intermediate stage, starting with a 16 × 16 block of intermediate values (M = 16), all pairs of consecutive values for columns/rows are added as shown in Fig. 6b. So, both the 16 × 8 (M × M/2) and the 8 × 16 (M/2 × M) intermediate blocks, each one with 128 SADs, correspond to 2N × N and N × 2N symmetric partitions of all possible 8 × 8 CUs contained in the current 64 × 64 CTU. This SAD aggregation process is followed until the last partition size is reached (1 × 1), i.e., the SAD in the last stage corresponding to the 2N × 2N partition of 64 × 64 CU (see Fig. 6c).

A particular case is the way asymmetric partitions are obtained from SADs corresponding to symmetric partitions. The idea is to repeat the same type of aggregation as the last one performed. If the start block has been obtained by the sum of consecutive columns, then the resulting consecutive columns are added again. The obtained values are SADs corresponding to asymmetric partitioning (left, right, up, and down) of the next size of CUs. For instance, in the last intermediate stage (M = 4, N = 16), after a sum of consecutive columns, we start with a 4 × 2 block of 8 SADs values corresponding to the 2N × N symmetric partition of the 4 32 × 32 CUs contained in the current 64 × 64 CTU. Then, all pairs of consecutive columns are added again as shown in Fig. 6b. Thus, a 4 × 1 block of SAD values are obtained corresponding to 2N × nU and 2N × nD asymmetric partitions of the current 64 × 64 CTU.

Thus, in the proposed architecture, the SATB module delivers 677 SADs of the current CTU block every single clock cycle to the next module, the comparison block.

3.3 Comparison block

The comparison block should keep the minimum SAD values for each CU partition with their corresponding motion vectors (search area positions). So, it will compare all incoming SADs from the SATB with the minimum SADs previously found. In a clock cycle, the comparison block receives 677 SADs corresponding to all partitions of all CUs contained in the current CTU, which is located in a particular position of the search area. So, in the next cycle, this module again receives 677 SADs corresponding to the next position of the search area. Therefore, this block compares SADs partition by partition, keeping the minimum SADs and the positions of the search area corresponding to those minimums. After comparing the SADs from the last search area location, the minimum SADs for each partition and the associated motion vectors are obtained.

4 Implementation results

The proposed architecture is designed as a pipeline process shown in Fig. 7. The memory reading process and shift registers propagation require only one clock cycle. The PUs use one cycle, the SATB requires twelve additional clock cycles, and the comparison block needs one additional clock cycle. So, the proposed architecture requires 63 clock cycles to perform the initial load of the shift registers, 15 clock cycles to load the pipeline, and then as many clock cycles as positions the search area has.

Our proposal has been modeled in VHDL, and it has been synthesized, simulated, and implemented on the Xilinx FPGA, Virtex-7 XC7VX550T-3FFG1158. The correctness of our design was tested and verified with the HEVC HM 14 reference model [17].

To evaluate the performance and efficiency of our design, we have parametrized our IME architecture to

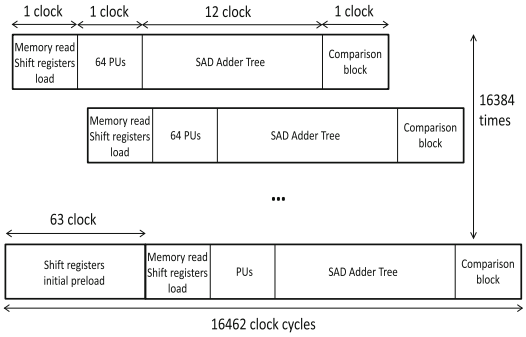


Fig. 7 Pipeline process of the proposed architecture

allow different configurations, such as (a) the maximum CTU size with values of 64 × 64 and 32 × 32, and (b) the size of the search area of the reference frame with values defined as the double size of the CTU, 80 % of the double size of the CTU, and the same size as a CTU.

Firstly, we proceed to test our proposal with the Virtex 7 FPGA technology. In Table 2, we show (a) the resulting operating frequency (clock), (b) the number of clock cycles for each CTU (latency), and (c) the system throughput in terms of the maximum frame rate under different video formats (1080p, 2K, and 4K), for different configurations of CTU and search area sizes. Our design can operate at the frequency of 247 and 318 MHz for a 64 × 64 CTU and a 32 × 32 CTU, respectively. It enables the encoder to carry out the IME process with a 64 × 64 CTU size and a search area of 128 × 128 pixels (as the HM14 reference model [17] establishes), obtaining a throughput of 30 fps at 2K video formats (2K@30fps). Our proposal is able to process video in real time for both 1080p and 2K resolutions in all tested configurations, and also with 4K video formats if the search area size is the same as the CTU size, as can be seen in Table 2.

In Tables 3 and 4, we show the resources used to implement our proposal for maximum CTU sizes of 64 × 64 and 32 × 32, respectively, on a Virtex-7 FPGA. In both tables, we show the resource usage of each block of the proposed architecture, as a resource usage profile. As can be seen, the slice area required by flip-flops and LUTs increases (≈×4) linearly with the increase of the maximum CTU size, as expected. In terms of flip-flops, the SATB is the block that uses the most amount of them (around 40 % of the total) in both configurations. This is due to the 12-stage pipeline design of the SATB. Moreover, calculating the distortion among pixels needs 50 % of the LUTs, due to the amount of subtractions in absolute value required in this process, being 1024 operations performed at each clock cycle. Regarding the required memory for storing the search area reference pixels, 36 and 9 kB memories are used in the case of a 64 × 64 CTU and a 32 × 32 CTU, respectively. On a Virtex-7, a BRAM block has a capacity of 36 kb. So, the slice area demanded by the used BRAMs also increases (≈×4) when going from 32 × 32 to the 64 × 64 maximum CU size.

An interesting analysis of our design can be observed at Table 2 when comparing the results of the 64 × 64 search area size with both CTU sizes. As can be seen, the latency is nearly the same but the throughput of the 64 × 64 CTU size is more than triple than the one obtained with the 32 × 32 CTU size. In terms of resource usage, the 64 × 64 CTU size requires near four times more resources, as shown in Tables 3 and 4. This implies that the use of more resources in the design provides higher throughput in a 4:3 relationship.

4.1 Systems evaluation

In Table 5, we compare our proposal with previous state-of-art architectures implemented on different FPGA platforms for both the 64 × 64 CTU and the 32 × 32 CTU size, and different search area sizes. We have chosen those works whose architectures are comparable to our proposal (i.e., perform the same functionality) and were implemented under FPGA technology. To make the comparison as fair as possible, we have obtained the performance results of our proposal with the same technologies, CTU sizes, and search area sizes as the ones used by the selected candidates. We will consider the system throughput as the main performance result of every proposal under comparison.

Regarding results for the 64 × 64 CTU size, Medhat et al. [3] present a parallel SAD block for the HEVC

Table 2 Throughput for different configurations in Virtex-7

CTU size Search area	64 × 64 128 × 128	64 × 64 104 × 104	64 × 64 64 × 64	32 × 32 64 × 64	32 × 32 52 × 52	32 × 32 32 × 32
Clock (MHz)	247	247	247	318	318	318
Latency	16,462	10,894	4174	4142	2750	1070
Fps at 1080p	32	48	124	39	59	151
Fps at 2K	30	45	116	37	55	141
Fps at 4K	8	12	30	10	15	37

Table 3 Utilization resources for 64 × 64 CTU implementation in Virtex-7

Resources	Flip-flops	LUTs	Memory (kB)
Memory read controller block	36,657 (25.40 %)	36,413 (19.30 %)	36 (100 %)
PUs (distortion computation)	32,768 (22.71 %)	94,208 (49.93 %)	–
SAD adder tree block (SATB)	58,727 (40.70 %)	47,063 (24.95 %)	–
Comparison block	16,150 (11.19 %)	10,980 (5.82 %)	–
Total	144,302	188,664	36

Table 4 Utilization resources for 32 × 32 CTU implementation in Virtex-7

Resources	Flip-flops	LUTs	Memory (kB)
Memory read controller block	10,155 (27.55 %)	9812 (20.22 %)	9 (100 %)
PUs (distortion computation)	8192 (22.22 %)	24,541 (50.57 %)	–
SAD adder tree block (SATB)	14,580 (39.55 %)	11,445 (23.58 %)	–
Comparison block	3937 (10.68 %)	2733 (5.63 %)	–
Total	36,864	48,531	9

Table 5 Comparison of the proposed architecture with state-of-the-art works

Design	Medhat [3]	Proposal 1	D'huys [7]	Proposal 2	Yuan [6]	Proposal 3
CTU size	64 × 64	64 × 64	64 × 64	64 × 64	32 × 32	32 × 32
Search area	104 × 104	104 × 104	64 × 64	64 × 64	48 × 48	48 × 48
Technology	Virtex-7	Virtex-7	Virtex-5	Virtex-5	Virtex-6	Virtex-6
Clock (MHz)	458.7	247	150	159	110	200
AMP	No	Yes	No	Yes	Yes	Yes
Throughput	2K@30fps	2K@45fps	720p@57fps	720p@173fps	1080p@30fps	1080p@43fps
Flip-Flops	39,901	144,302	199,682	178,620	19,744	43,531
LUTs	24,957	188,664	210,158	184,288	55,346	45,752
Memory (kB)	44	36	1229	36	148	9

integer-pel full search architecture without supporting AMP modes with a search area of 104 × 104 pixels. They used the Virtex-7 technology, and their design can operate at the frequency of 458.7 MHz. The operating frequency of our proposal with the same technology and configurations is almost two times lower. However, our architecture is capable of processing 45 fps at 2K video formats instead of 30 fps as obtained by the proposed design in [3]. Therefore, our proposed architecture is 1.5× as fast as the one proposed in [3] using the same search area size and considering all the AMP partition modes, contrary to [3], where AMP partitions are not calculated. This is due to the fact that our design takes advantage of the minimal latency to perform the same operations as we have an efficient pipeline design. Therefore, our system achieves higher throughput, reaching real-time processing for 2K video resolutions at 45 fps, and being on the way to accomplishing the same goal for 4K video formats, where 12 fps were obtained.

On the other hand, D'huys [7] proposes a reconfigurable design for HEVC motion estimation which can operate at the frequency of 150 MHz. His architecture is compared with our proposal, setting a common search area size to 64 × 64 pixels and the Virtex-5 technology. The operation frequency of our proposal is 159 MHz, achieving system throughput of 20 fps at 4K and 75 fps at 2K video formats. Our design significantly improves the performance of the architecture presented in [7], which is able to process a lower resolution video (720p) at 57 fps. If the video resolution is set to 720p, our architecture is capable of processing 173 fps. So, our architecture presents good balance between the maximum frequency and pipeline processing design, taking advantage of the low latency by leveraging all available resources.

Regarding results for the 32 × 32 CTU size, in Table 5, we show the comparison results between our proposal (implemented on a Virtex-6 FPGA) and the integer motion estimation design found in [6], both with a search area size of 48 × 48 pixels. The most significant feature, worthy of attention, is that our proposal can provide a higher operation frequency, achieving throughput of 43 fps at 1080p and 40 fps at 2K resolution, whereas the architecture presented in [6] is able to achieve 30 fps at 1080p video formats, using a similar amount of FPGA resources.

Considering the presented results, our architecture shows an efficient implementation of available resources in

FPGA, overcoming the performance of previous state-of-the-art architectures.

4.2 HEVC R/D performance and time profiling

To better understand the capabilities of IME hardware devices, we have performed a set of tests to analyze the benefits of including an IME FPGA-based accelerator, like the one proposed here, in the HEVC reference software in terms of speedup and observe how both parameters, the CTU size, and the search area size impact on the R/D performance of the HEVC encoder. To perform these tests, we have used the HEVC HM 14 reference model [17] working with the main profile and low-delay configuration mode. The HEVC reference software was compiled with Visual Studio 2010 with the default compiler options and run over a PC platform with an Intel Core i7-3770 CPU 3.40 GHz with 16 GB RAM. Three video sequences from the HEVC common conditions video set were selected: (s1) Racehorses at 832 × 480 resolution (30 fps), (s2) Basketball Drive at 1920 × 1080 (50 fps), and (s3) People On Street at 2560 × 1600 (30 fps).

The experiments were performed using different search area sizes (128 × 128, 104 × 104, 64 × 64, 52 × 52, and 32 × 32) and CTU sizes (64 × 64 and 32 × 32).

Tables 6 and 7 show all data gathered for CTU sizes of 64 × 64 and 32 × 32, respectively. The first row shows the total time (in seconds) required to encode each video sequence (10 frames). The second row shows the percentage of the total time needed by the IME software module using a full search algorithm (% IME time SW). These percentages vary from 62 to 96 % depending on the video sequence, the search area size, and the CTU size. As was expected, the time required by the IME software module decreases as both the search area size and the CTU size do. Rows three and four show the number of CTUs per second that can be computed by software (CTU/s SW) and hardware (CTU/s HW) versions of the IME module. As can be seen, these values also depend on the search area size and maximum CTU size, and in the case of the IME software module, also depend on the video sequence.

So by looking at the information provided in Tables 6 and 7, we could assess that the IME module is a bottleneck in the HEVC reference software. Therefore, if the IME software module is replaced by our FPGA-based device, the overall encoding time will be significantly reduced. For example, for a high-resolution video sequence like PeopleOnStreet (s3) and setting the CTU size to 64 × 64 and the search area size to 128 × 128 (default values in the HEVC reference software), the total encoding time (10 frames) will be reduced from 38 h to 2 s, since the motion estimation module takes around 95 % of the overall encoding time.

To reduce the hardware complexity, allowing faster versions with reduced power consumption, the CTU size and the search area must be reduced as much as possible. However, this may cause performance degradation in the encoding process, decreasing the overall video quality and/or reducing the compression rate. To evaluate this aspect, we will analyze the impact of these parameters on the R/D (rate/distortion) performance. In Fig. 8, we show the video quality of the test video sequence RaceHorses (s1) for each CTU and search area sizes at different compression levels (QP values). As can be seen, there are slight differences between the CTU size, being greater the difference as the compression rate increases. Differences between search areas are negligible. Although R/D differences may depend on the video content, similar results were obtained for the other two video sequences tested.

Finally, we also have performed a profile of the IME HEVC with another motion search algorithm, which is available in the HEVC reference software (diamond-like search). This algorithm is used by default in the reference software and it is about 90 times as fast as the full search algorithm, with the disadvantage that it does not guarantee finding optimal MVs, and as consequence video quality could be affected. As can be seen in Table 8, the inclusion of our IME hardware module will speed up the IME computation of diamond-like search algorithm 230 and 700 times for 32 × 32 and 64 × 64 CTU sizes, respectively.

After performing the whole analysis, a trade-off should be taken to determine which configuration better adapts to the application requirements (low power consumption,

Table 6 Time profile of the IME HEVC for a 64 × 64 CTU

Search area	128 × 128			104 × 104			64 × 64		
Video sequences	s1	s2	s3	s1	s2	s3	s1	s2	s3
Encoding time SW (s)	13,670	61,602	135,970	9392	42,050	92,863	4117	17,881	39,933
% IME time SW	95	96	95	90	94	93	83	86	85
CTU/s SW	0.24	0.26	0.23	0.38	0.39	0.35	0.91	1.00	0.89
CTU/s HW	14,993	14,993	14,993	22,625	22,625	22,625	59,172	59,172	59,172
HW gain	62,260	57,767	64,800	59,621	58,312	65,384	64,856	59,115	66,477

Table 7 Time profile of the IME HEVC for a 32 × 32 CTU

Search area	64 × 64			52 × 52			32 × 32		
Video sequences	s1	s2	s3	s1	s2	s3	s1	s2	s3
Encoding time SW (s)	3652	15,892	35,295	2634	11,303	25,293	1378	5748	12,911
% IME time SW	85	87	86	79	81	81	60	63	62
CTU/s SW	4	5	4	6	7	6	14	17	15
CTU/s HW	76,923	76,923	76,923	115,607	115,607	115,607	297,619	297,619	297,619
HW gain	20,383	17,293	19,457	20,654	17,384	19,675	21,096	17,664	19,912

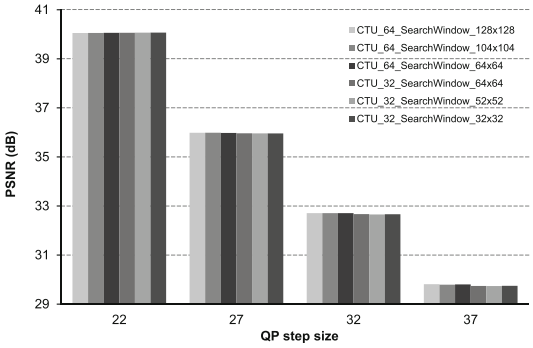


Fig. 8 R/D performance for different CTU and search area sizes for the RaceHorses sequence

Table 8 Average CTU IME time with 2 × CTU search area size

CTU size	Full search SW	Diamond search SW	Full search HW
64 × 64	4.11 s	4.65E−02 s	6.67E−05 s
32 × 32	2.48E−01 s	3.05E−403 s	1.30E−05 s

encoding time, compressed video quality). The use of hardware accelerators designed in FPGA platforms like the one proposed here are mandatory when real-time UHD video encoding is the objective.

5 Conclusion

In this work, we have presented a fast and efficient IME hardware unit for the HEVC video encoder which (a) supports AMP modes, (b) both CTU and search area sizes are configurable, and (c) is implemented on a Virtex-7 FPGA. The suitability of using FPGAs for implementing the HEVC IME module has been demonstrated in this paper, proposing a design that improves the previous results of other IME hardware systems.

Our FPGA-based design is able to process 2K video formats at 116 frames per second and 4K video sequences at 30 fps, which represents a huge complexity reduction of the HEVC video encoding process, achieving real-time encoding for high-definition video contents and beyond.

We have also analyzed the impact of the maximum CTU and the search area sizes over the encoder complexity and the resulting video quality, showing that the encoder complexity decreases as both the maximum CTU size and the search area do. Furthermore, the maximum CTU size has a minimum impact over the R/D, being more noticeable at high compression rates. In the test video sequences analyzed, the impact over the quality of the search are size is negligible, but it will depend on the video content.

In future work, we are working to include our IME hardware module in the HEVC reference software and perform a complete test over an evaluation platform such as ZYNQ of Xilinx. In addition, we intend to expand the hardware module to perform the fractional-pel motion estimation, or even the SAD unit for intra-mode coding.

Acknowledgments This research was supported by the Spanish Ministry of Economy and Competitiveness under Grant TIN2015-66972-C5-4-R.

References

1. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. **22**, 1649–1668 (2012)
2. Sze, V., Budagavi, M., Sullivan, G.J.: High Efficiency Video Coding (HEVC) Algorithms and Architectures. Springer, Switzerland (2014)
3. Medhat, A., Shalaby, A., Sayed, M.S., Elsabrouty, M.: A Highly Parallel SAD Architecture for Motion Estimation in HEVC Encoder. In: IEEE Asia Pacific Conf. Circuits Syst. (APCCAS), pp. 280–283. Ishigaki (2014)
4. Byun, J., Jung, Y., Kim, J.: Design of integer motion estimator of HEVC for asymmetric motion-partitioning mode and 4K-UHD. Electron. Lett. **49**(18), 1142–1143 (2013)
5. Vidyalekshmi V.G., Yagain D., Ganesh Rao K.: Motion estimation block for HEVC encoder on FPGA. In: IEEE Int. Conf.

Recent Advances and Innovations in Engineering (ICRAIE), pp. 1–5. Jaipur, (2014)
6. Yuan, X., Jinsong, L., Liwei, G., Zhi, Z., Teng, R.K.F.: A high performance VLSI architecture for integer motion estimation in HEVC. In: IEEE 10th Int. Conf. ASIC (ASICON), pp. 1–4. Shenzhen (2013)
7. D'huys, T.: Reconfigurable data flow engine for HEVC motion estimation. In: IEEE Int. Conf. Image Processing (ICIP), pp. 1223–1227. Paris (2014)
8. Davis, P., Sangeetha, M.: Implementation of motion estimation algorithm for H.265/HEVC. Int. J. Adv. Res. Elect. Electron. Instrum. Eng. **3**(3), 122–126 (2014)
9. Nalluri, P., Alves, L.N., Navarro, A.: High speed SAD architectures for variable block size motion estimation in HEVC video coding. In: IEEE Int. Conf. Image Processing (ICIP), pp. 1233–1237. Paris (2014)
10. Chen, C.Y., Chien, S.Y., Huang, Y.W., Chen, T.C., Wang, T.C., Chen, L.G.: Analysis and architecture design of variable block-size motion estimation for H.264/AVC. IEEE Trans. Circuits Syst I: Reg. Papers **53**(3), 578–893 (2006)
11. Elhamzi, W., Dubois, J., Miteran, J.: An efficient low-cost FPGA implementation of a configurable motion estimation for H.264 video coding. Springer J. Real-Time Process. **9**(1), 19–30 (2014)
12. Moorthy, T., Ye, A.: A scalable architecture for variable block size motion estimation on field-programmable gate arrays. In: IEEE Canadian Conf. Electrical and Computer Engineering (CCECE), pp. 1303–1308. Niagara Falls (2008)
13. Kthiri, M., Kadionik, P., Levi, H., Loukil, H., Atitallah,,B., Masmoudi, N.: An FPGA implementation of motion estimation algorithm for H.264/AVC. In: IEEE 5th Int. Symp. I/V Communications and Mobile Network (ISVC), pp. 1–4. Rabat (2010)
14. Pastuszak, G., Jakubowski, M.: Adaptive computationally scalable motion estimation for the hardware H.264/AVC encoder. IEEE Trans. Circuits Syst. Video Technol. **23**(5), 802–812 (2013)
15. Pastuszak, G., Trochimiuk, M.: Algorithm and architecture design of the motion estimation for the H.265/HEVC 4K-UHD encoder. J. Real Time Image Process (2015)
16. Lin, Y.L.S., Kao, C.Y., Kuo, H.C., Hen, J.W.: VLSI Design for Video Coding-H.264/AVC Encoding from Standard Specification to Chip. Springer, New York (2010)
17. HEVC software repository HM–14.0 reference model. https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-14.0. Accessed 2 May 2014 (2014)

Estefania Alcocer was born in Bigastro, Spain, in 1986. She received her M.S. degree in telecommunication engineering in 2010 from the Miguel Hernandez University, Elche, Spain, and she joined the GATCOM research group as Ph.D. student in 2012. Currently, she is an assistant professor in the Department of Physics and Computer Architecture at Miguel Hernandez University, Elche since 2012. Her current research activities are related to image processing, the design of FPGA-based systems and video coding.

Roberto Gutierrez was born in Orihuela, Spain, in 1977. He received his M.Sc. degree in telecommunication engineering in 2003, and the Ph.D. degree in electronic engineering in 2011, both from the Universidad Politecnica de Valencia, Spain. He is an associate professor in the Department of Communication engineering at Universidad Miguel Hernandez, Elche since 2003. His current research interests include the design of FPGA-based systems, computer arithmetic, VLSI signal processing and digital communications.

Otoniel Lopez-Granado received his M.S. in Computer Science from the University of Alicante (Spain) in 1996. Between 1997 and 2003 he worked as programmer analyst in an important industrial informatics firm. In 2003, he joined to the Computer Engineering Department at Miguel Hernandez University (UMH), Spain, as an assistant professor. Then, he received the Ph.D. degree in Computer Science in 2010. In 2012, he was promoted to associate professor. Currently, he leads the GATCOM research group (atc.umh.es) at Miguel Hernandez University. His research and teaching activities are related to multimedia networking (audio/video coding and network delivery).

Manuel P. Malumbres received his B.Sc. in Computer Science from the University of Oviedo (Spain) in 1986. In 1989, he joined to the Computer Engineering Department (DISCA) at Technical University of Valencia (UPV), Spain, as an assistant professor. Then, he received M.S. and Ph.D. degrees in Computer Science from UPV, in 1991 and 1996, respectively. He is a TC member of IEEE Multimedia Communications Group and associate editor of the Signal, Image and Video Processing journal. He was serving as TPC member of several relevant international Conferences related with his main research interests. He is author of more than 160 conference and journal publications and several networking books for undergraduate CS courses. Currently, his research and teaching activities are related to multimedia networking (image/video coding and network delivery) and wireless network technologies (MANETs, VANETs and WSNs).

EURASIP Journal on Advances in Signal Processing
a SpringerOpen Journal

RESEARCH Open Access

MPCM: a hardware coder for super slow motion video sequences

Estefanía Alcocer[1*], Otoniel López-Granado[1], Roberto Gutierrez[2] and Manuel P Malumbres[1]

Abstract

In the last decade, the improvements in VLSI levels and image sensor technologies have led to a frenetic rush to provide image sensors with higher resolutions and faster frame rates. As a result, video devices were designed to capture real-time video at high-resolution formats with frame rates reaching 1,000 fps and beyond. These ultrahigh-speed video cameras are widely used in scientific and industrial applications, such as car crash tests, combustion research, materials research and testing, fluid dynamics, and flow visualization that demand real-time video capturing at extremely high frame rates with high-definition formats. Therefore, data storage capability, communication bandwidth, processing time, and power consumption are critical parameters that should be carefully considered in their design. In this paper, we propose a fast FPGA implementation of a simple codec called modulo-pulse code modulation (MPCM) which is able to reduce the bandwidth requirements up to 1.7 times at the same image quality when compared with PCM coding. This allows current high-speed cameras to capture in a continuous manner through a 40-Gbit Ethernet point-to-point access.

Keywords: PCM; Image coding; FPGA design; High speed; Integrated circuits

1 Introduction

Video compression has been an extremely successful technology that has found its commercial application across many areas from scientific and industrial applications as video archiving, high-quality medical video, surveillance and security applications to the audiovisual industry (TV and cinema) and the broad spectrum of video appliances available in the market, such as digital cameras, DVD, Blue-Ray, and DVB.

In the last decade, the improvements in VLSI levels and image sensor technologies have led to a frenetic rush to provide image sensors with higher resolutions and faster frame rates. As a result, video devices were designed to capture real-time video at high-resolution formats with frame rates above 100 Hz. Nowadays, ultrahigh-speed video cameras can be found in the market like Phantom v641 (Vision Research Inc., Wayne, NJ, USA) [1] which is able to capture high-resolution video ($2,560 \times 1,600$ pixels) at 1,450 frames per second (fps). These video cameras are specially suited for scientific or industrial applications, such as car crash tests, explosives and pyrotechnics, ballistics, projectile tracking, combustion research, materials research and testing, fluid dynamics, flow visualization. which demand real-time video capturing at extremely high frame rates with high-definition (HD) formats. Therefore, data storage capability, communication bandwidth, processing time and power consumption are critical parameters that should be carefully considered in the design process of high-speed video cameras.

In order to fight against these constraints, most of nowadays high-speed cameras store the captured images in a fast synchronous dynamic random access memory (SDRAM) module of up to 64 GB [1-4] without performing compression, using pulse code modulation (PCM) [5]. The huge amount of data of the resulting uncompressed image/video needs to be processed to guarantee its transmission or storage, being a really challenging task. Thus, the internal communication bus may not be fast enough to transfer the video out of the camera, or the writing speed of the storage device may not be high enough to save the video [6]. So the approach of using fast SDRAM memory as video storage is feasible since the memory bandwidth is high enough, but when memory is run out, the camera

*Correspondence: ealcocer@umh.es
[1]Physics and Computer Architecture Department, Miguel Hernández University, Elche 03202, Spain
Full list of author information is available at the end of the article

stops recording and needs to save the stored video to a secondary storage in raw or compressed format. This is a limitation because depending on the capturing resolution of the camera, only a few seconds could be recorded in the random access memory (RAM) module, and so continuous capturing is not possible.

So as to overcome these restrictions, it would be of interest to reduce the video storage requirements by means of hardware encoders that fulfill the application requirements, i.e., high frame rate and high-definition and beyond video formats. Therefore, if we were able to perform some kind of ultrafast encoding, we would reduce the required storage resources, and real-time recording like in conventional video cameras would be possible.

Many hardware coders based on different coding algorithms are used in real systems [7-14]. Most of them are application-specific integrated circuits dedicated to specific encoding algorithms that are not designed to work in real-time with ultrahigh frame rates and high-definition video formats.

However, several attempts have been made in order to deal with high-speed camera encoding. In [15] authors present JPEG field-programmable gate array (FPGA)-based encoder which is able to compress up to 500 frames/s at a resolution of $1,280 \times 1,024$. Also in [16], an improved version of the fast boundary adaptation rule [17] algorithm in conjunction with differential pulse code modulation is applied to increase the R/D efficiency, although coding delays were not provided.

In general, the constraints imposed by ultrahigh frame rate video capture applications discard most of the existing coding techniques (e.g., predictive coding or transform coding) since they are much more complex than PCM. Therefore, a coding algorithm that has properties similar to those of PCM (low complexity, random access, and scalability) but with a better coding efficiency would be of interest. Modulo-Pulse Code Modulation scheme (MPCM) [18] image coder fulfills these requirements. To encode an image, MPCM encoder removes certain bits from each pixel value which represents a very simple processing. The complexity is moved to the decoder side, where the bits that were removed from each pixel will be predicted by using its codeword (remaining bits of a pixel) and side information (SI) that the decoder computes by interpolating the previously decoded pixels.

In this paper we implement a fast codec based on MPCM [18] over a XC7Z020-1CLG484CES Xilinx FPGA device (Xilinx Inc., San Jose, CA, USA). Results show that FPGA-based MPCM encoder obtains a throughput of up to 409.84 MBytes/s at high compression rates for monochromatic images, allowing to store on a nonvolatile memory 2,501 fps at a $1,280 \times 1,024$ resolution. Furthermore, in this paper we present a hardware implementation of the MPCM decoding system, which is able to reproduce a full-HD (1080p) video at 193 fps.

The rest of the paper is organized as follows. In Section 2 we present a brief overview of the Modulo-PCM encoder. In Section 3, the description of the proposed architecture is presented. A detailed evaluation of the architecture proposal is shown in Section 4 in terms of R/D, coding delay, power consumption, and occupied board area. Finally, in Section 5 some conclusions are drawn.

2 Encoding system

In this section, we describe the MPCM-based coding algorithm for the encoding of a one-dimensional signal. Let $\widetilde{x}_n$ ($n \in N$) be a continuous-amplitude discrete-time signal whose amplitude values lie in $[A_{\min}, A_{\max}]$. Let x_n be the digital signal that results from the quantization of $\widetilde{x}_n$ with a fixed-rate uniform quantizer of B bits/sample and step size:

$$\Delta = (A_{\max} - A_{\min})/2^B.$$

The easiest way of reducing the bit rate of $\widetilde{x}_n$ is to remove the l-least significant bits (LSBs) of each codeword of $\widetilde{x}_n$. To achieve a more efficient rate reduction, we propose the use of a MPCM-based coding algorithm (Figure 1). The samples of $\widetilde{x}_n$ are divided into sets that are encoded with different accuracies. For the sake of simplicity, let us consider we divide $\widetilde{x}_n$ into two sets: $S_0 = \{x_{2n} | n \in i = 0, 1, 2, \ldots\}$ and $S_1 = \{x_{2n+1} | n = 0, 1, 2, \ldots\}$. As shown in Figure 1, each sample in S_0 is encoded by removing the l_0-LSBs of its codeword (PCM signal) while each sample in S_1 is encoded by removing the m_1-most significant bits (MSBs) and l_1-LSBs of its codeword (MPCM signal). Then, the encoder works at an average rate $R = B - (l_0 + l_1 + m_1)/2$ bits/sample.

As the encoding of x_{2n} is equivalent to a quantization with a uniform quantizer with step size equal to $2^{l_0}\Delta$, the decoder can directly reconstruct the samples of S_0 from their codewords (Figure 1). With respect to the encoded samples in S_1 (Figure 2 (a)), removing the l_1-LSBs of x_{2n+1} is equivalent to quantizing its original continuous value with a uniform quantizer with step size equal to $2^{l_1}\Delta$ (Figure 2 (b)). After removing the m_1-MSBs, the resulting codeword identifies a set of 2^{m_1} disjoint intervals $\{I_i | i = 0, \ldots, 2^{m_1} - 1\}$, with each interval being of length $2^{l_1}\Delta$ (Figure 2 (c)).

At the decoder, a PCM-coded signal is directly reconstructed from its received codeword $\hat{x}_{2n}$. This reconstructed value is set to the midpoint of its quantization interval. However, a MPCM-coded signal is decoded by using its codeword $\hat{x}_{2n+1}$ and its SI y_{2n+1}. This process is divided into two steps: decision and reconstruction. Consequently, in order to decide which I_i interval $\hat{x}_{2n+1}$ belongs to, MPCM decoders exploit the correlation between the signal samples by furnishing a prediction for

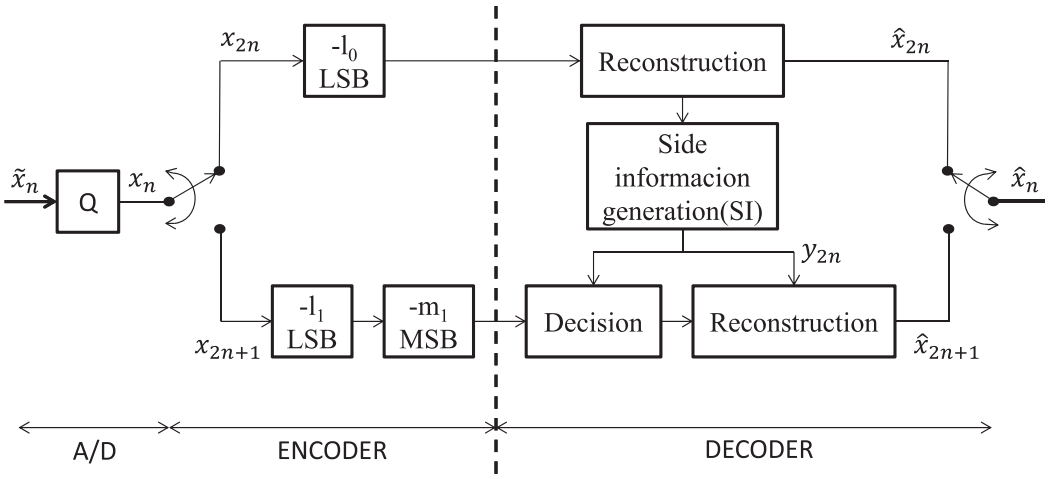


Figure 1 Block diagram of MPCM coding algorithm.

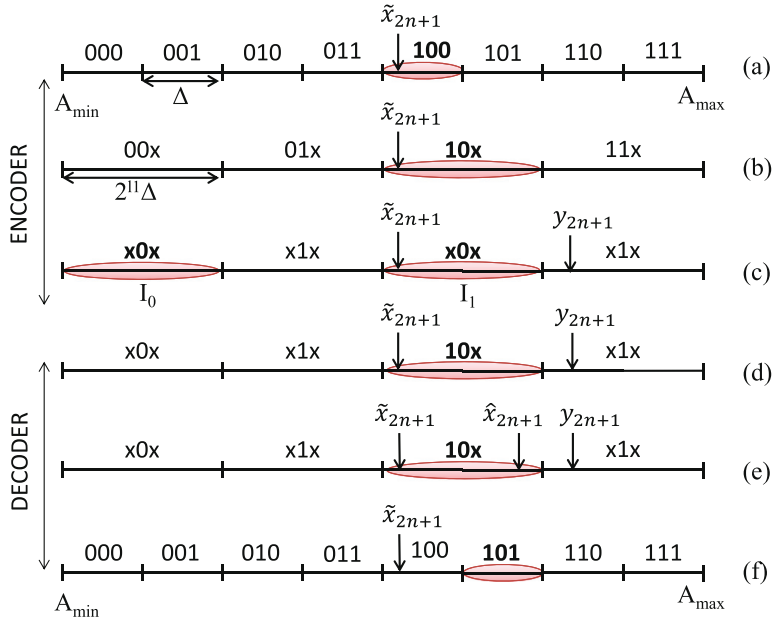


Figure 2 Quantization intervals and codewords for $B = 3$, $l_1 = 1$ and $m_1 = 1$. (a) After A/D conversion. **(b)** After removing l_1 bits. **(c)** After removing m_1 bits. **(d)** After deciding between l_0 and l_1. **(e)** After reconstruction. **(f)** After final quantization. Symbol X represents the removed bits. Boldfaced codewords represent the codeword selected in each step for the shown values of x_{2n} and y_{2n}. Marked intervals are the intervals represented by the selected codeword in each step.

each sample in S_1 based on previously decoded samples in S_0. This prediction y_{2n+1} acts as SI to decide the interval, which is obtained by interpolating the previously decoded samples in S_0. Therefore, in this decision step, the decoder uses y_{2n+1} to select one of the 2^{m_1} disjoint intervals as shown in (Figure 2 (d)), choosing the closest interval to the prediction. If the decision process is done without error for x_{2n+1}, its m_1-MSBs are properly recovered; otherwise, the decoder incurs a decision error in which the probability depends on m_1 and the accuracy of the SI. The accuracy of the SI depends on the degree of correlation between the samples x_n and the distortion introduced in the encoding of S_0. Furthermore, the larger the m_1, the shorter the minimum distance between the codewords of the same set 2^{m_1}, hence, the higher the probability of decision error. As a result, in order to limit these impacts in the encoding algorithm, l_0 must be lower than or equal to l_1 ($l_0 \leq l_1$) and m_1 must be the minimum possible. Once the decoder has estimated the m_1-MSBs of x_{2n+1} (Figure 2 (d)) in the reconstruction step, it tries to recover its l_1-LSBs which finally provides an estimated signal $\hat{x}_{2n+1}$ (Figure 2 (e)). This reconstruction is done using the quantization interval I_i where supposedly lies x_{2n+1} and its SI (y_{2n+1}) by taking the closest value of the chosen interval to the prediction y_{2n+1} (Figure 2 (f)). Notably, if the coding parameters accomplish the following characteristics $l_0 = l_1$ and $m_1 = 0$, the MPCM encoder/decoder will act as a PCM coding system, since in such cases, the help provided by the SI does not compensate the loss suffered by encoding each MPCM signal with fewer bits than the PCM signal. In fact, in these cases, midpoint reconstruction performs better than MPCM reconstruction. Nevertheless, in most cases, MPCM performs better than or the same as PCM, with great gains at 1, 2, 3, and 4 bpp.

For a more detailed description of MPCM encoder, the reader is referred to [18,19].

3 Hardware implementation

In order to cope with the high throughput bandwidth of nowadays high-speed cameras, both MPCM encoder and decoder have been implemented over a hardware architecture. The description language used to build its design is VHDL. The proposed hardware implementation has been developed over a Zynq-7000 FPGA of Xilinx family, specifically over the ZC702 model which includes the XC7Z020-1CLG484CES SoC (System-on-Chip) [20].

3.1 Encoder architecture

The implemented encoder architecture is illustrated in Figure 3. The original image/frame captured by the camera sensors is stored in a memory block whose reading is determined by a control block. In that structure, 16 pixels are read on each cycle so as to speed the encoding process as much as possible within the scope of the device's internal memory. This memory acts as an internal buffer of frames to read them in high-speed applications and it is implemented with block RAMs. In this way, we use 52 dual-port 36-Kb block RAMs with ports configured as 512×64 bits, where 64 output bits, namely 8 pixels, are read in each memory output port. These pixels are processed in the following block, without delay, in the same reading cycle, where they are encoded by removing the corresponding l_0 or l_k and m_k bits. Finally, we obtain the coded samples of the image which are sent to the final storage device.

Our proposed implementation design first divides the image/frame into N set of pixels, where one of the resulting pixels is encoded using PCM and the rest of them with MPCM. We take the strategy of dividing $x[n_1, n_2]$ into

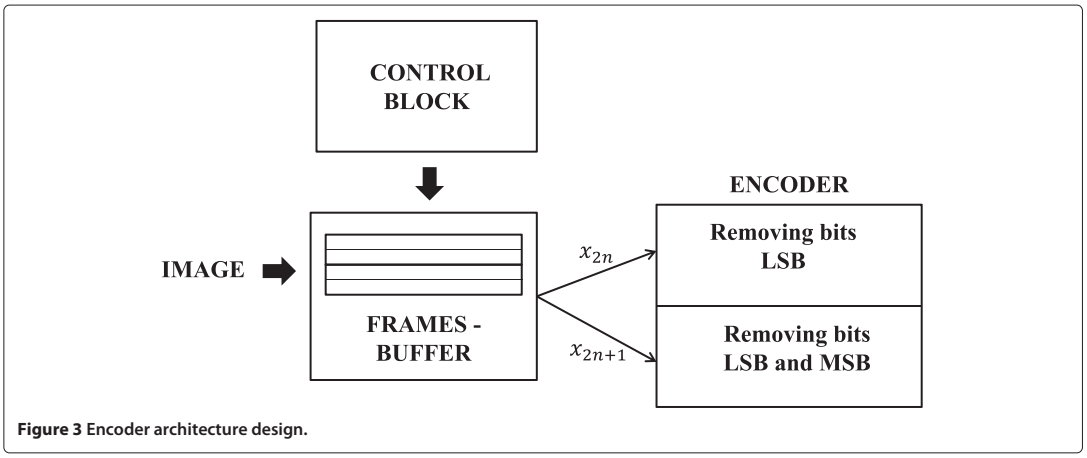


Figure 3 Encoder architecture design.

four ($N = 4$) sets $x_{0,0}[n_1, n_2], x_{0,1}[n_1, n_2], x_{1,0}[n_1, n_2]$, and $x_{1,1}[n_1, n_2]$ such that

$$x_{p,q}[n_1, n_2] = x\left[2n_1 + p, 2n_2 + q\right]$$

with p and $q \in [0, 1]$ as explained in [19]. Then, the first one is encoded using PCM, removing l_0 LSBs, and the remaining parts using MPCM, removing l_k LSBs and m_k MSBs. A diagram of the steps used in our hardware algorithm is shown in Figure 4. Remark that both the reading and coding of the 16 pixels are performed in the same clock cycle.

3.2 Decoder architecture

The proposed decoder architecture is shown in Figure 5. In this approach, the buffer is filled with coded samples until the first three lines are completed since at least three rows and three columns of pixels stored are needed (9 pixels in total) so as to decode a set N of 4 pixels, then the decoding process starts. The decoder is divided into two steps: decision and reconstruction. The recovery of the original image is performed as follows:

1. Three columns of data are processed in decision block, where the PCM signal is reconstructed as shown in Figure 6a. SIs are generated from the previous PCM samples (Figure 6b), and one of the possible intervals 2^{m_k} is selected in which each MPCM decoded signal will be placed (Figure 6c).
2. At the decoder reconstruction block, we recover the LSB removed in the encoding process using its SI and the interval corresponding to each MPCM sample as shown in Figure 6d, according to this rule:

$$\hat{x} = \begin{cases} a & \text{if } y < a \\ y & \text{if } a \le y \le b \\ b & \text{if } y > b \end{cases}.$$

   After completing these steps, we get four decoded pixels.
3. In each cycle, two columns of encoded samples leave the decoder and another two enter into it, keeping the last previous column, considering that it contains the necessary PCM samples to decode the next set of 4 pixels. This process is carried out iteratively until all samples of the three rows from the buffer have been decoded.
4. Then, the buffer is shifted. Two rows leave the buffer and two new ones enter into it, keeping the last previous row, as in case of columns. All the above operations continue until all the input encoded samples are decoded.

The proposed decoder design has been completely pipelined, so this makes each of the aforementioned steps used for decoding to be performed concurrently. In addition, as previously mentioned, four decoded pixels are obtained in each cycle. In this approach, the operating

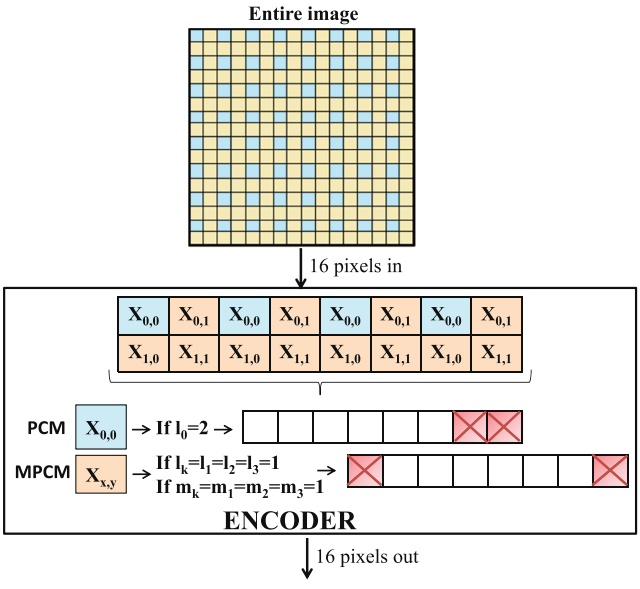


Figure 4 Example of the encoder algorithm with $l_0 = 2$, $l_k = 1$ and $m_k = 1$. Blue-colored blocks are samples to be coded using PCM and the yellow/orange ones are to be coded using MPCM.

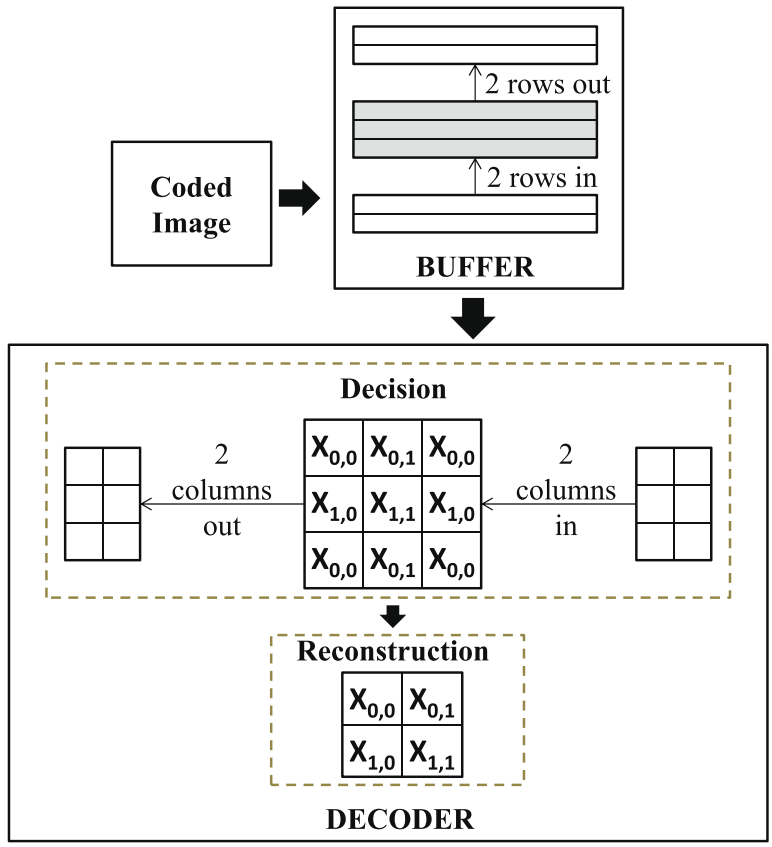


Figure 5 Decoder architecture design.

frequency has been set to get the 4 pixels, but in order to organize the entire decoded image, a phase-locked-loop (PLL) module has been used, which generates multiple clocks for a given input clock. Therefore in each cycle, 4 pixels are stored in a buffer with a fixed frequency, but these pixels are read with a frequency four times higher, thus achieving a serial output without delay. The buffers used in the proposed architecture have been implemented using single dual-port 18-Kb block RAMs.

4 Results

Both encoder and decoder architecture designs have been tested. In this section, we present the performance evaluation of the complete system in terms of peak signal-to-noise ratio (PSNR), encoding/decoding times, board area usage, maximum frame rate, and speed-ups obtained when compared to a CPU sequential algorithm. The architectures have been synthesized, placed and routed using Xilinx ISE 14.3 tool, and have been simulated and verified using Matlab/Simulink through System Generator toolbox. They have been designed into the Zynq AP SoC-based board previously mentioned. Occupied board area, maximum frequency, and power consumption estimation have been measured from the Xilinx ISE 14.3 tool. In our experiments, we have assessed the results of eight gray-scale images with 8 bits per sample, five of which (*Zelda, Lena, Peppers, Barbara, and Baboon*) with a resolution of 512×512 pixels and the rest, a full-HD (1080p) image, a $2,048 \times 2,560$ image, and a 4K UHD image. Furthermore, we have assigned values to the coding/decoding parameters (with $N = 4$, $l_1 = l_2 = l_3$, and $m_1 = m_2 = m_3$) so as to obtain the best result on average of PSNR for a given bit-rate, although there may be other combinations of parameters that would optimize a particular image as proposed by Marleen Morbee in [19].

In Table 1 the PSNR obtained for all tested images as a function of the bit rate (R) is presented. As expected, for

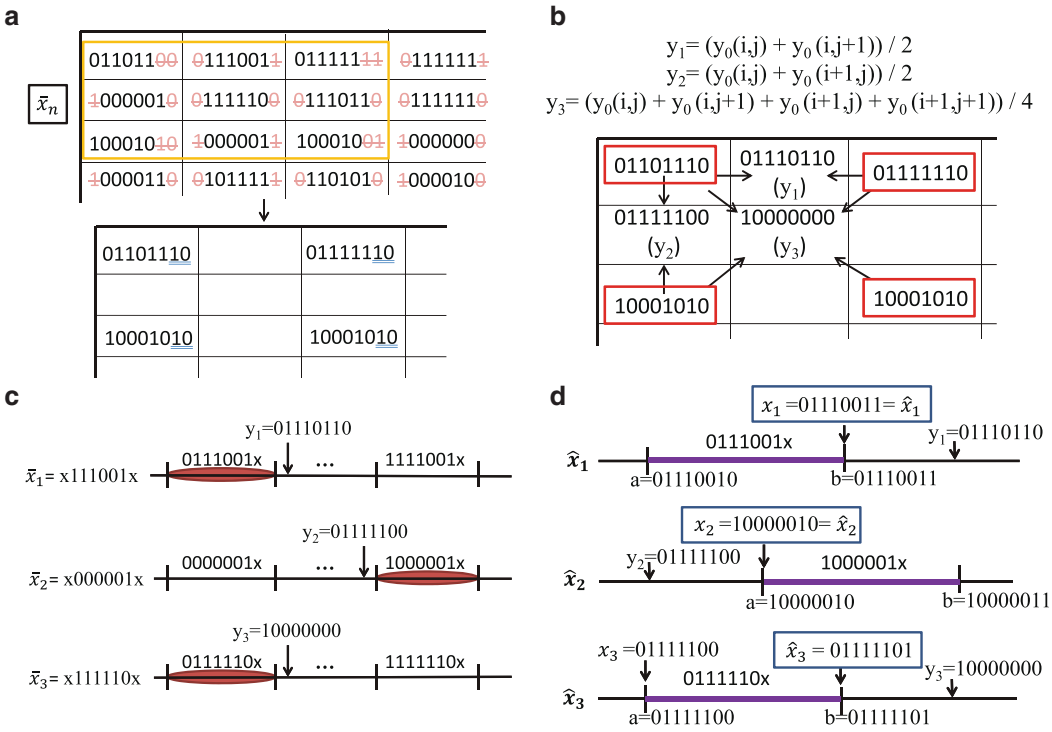


Figure 6 Example of MPCM decoding process with $l_0 = 2, l_1 = 1$ and $m_1 = 1$. (a) Reconstruction of PCM signals by taking the midpoint of its quantization interval. **(b)** Obtaining of SI (y_k) by interpolating the previously decoded PCM signals. **(c)** Decision step to recover MPCM signals. Interval selection of the 2^{m_1} potential codewords. **(d)** Reconstruction step to recover MPCM signals.

higher rates (R), which means removing few bits in the encoding process, MPCM algorithm generally provides good PSNR due to the fact that no significant loss occurs in the coding process, and, consequently, no big errors are introduced in the decoding process. Therefore, the lower the rate (R), the lower the PSNR value. Moreover, as explained in Section 2, in the cases where the chosen parameters meet $l_0 = l_1$ and $m_1 = 0$, a reconstruction PCM is applied, so the PSNR corresponding to $R = 6$ bpp and $R = 5$ bpp, shown in Table 1, is the same for

Table 1 PSNR values for all tested images for a given bit-rate

	PSNR (dB)				
	$R = 4$ bpp	$R = 4.5$ bpp	$R = 5$ bpp	$R = 5.5$ bpp	$R = 6$ bpp
Image	(l_0,l_1,m_1)	(l_0,l_1,m_1)	(l_0,l_1,m_1)	(l_0,l_1,m_1)	(l_0,l_1,m_1)
	(1,4,1)	**(2,4,0)**	**(3,3,0)**	**(1,3,0)**	**(2,2,0)**
Zelda (512 × 512)	39.00	38.90	40.15	41.51	45.04
Lena (512 × 512)	37.74	37.77	39.82	41.06	44.96
Peppers (512 × 512)	33.70	36.92	39.32	40.29	44.62
Barbara (512 × 512)	26.06	35.27	38.91	39.83	44.56
Baboon (512 × 512)	24.45	33.02	37.61	38.20	43.85
Tractor (1,920 × 1,080)	38.01	39.67	40.22	42.15	44.73
Woman (2,048 × 2,560)	30.53	36.47	39.52	40.70	44.95
Ducks (3,840 × 2,160)	35.09	35.00	38.14	38.88	43.72

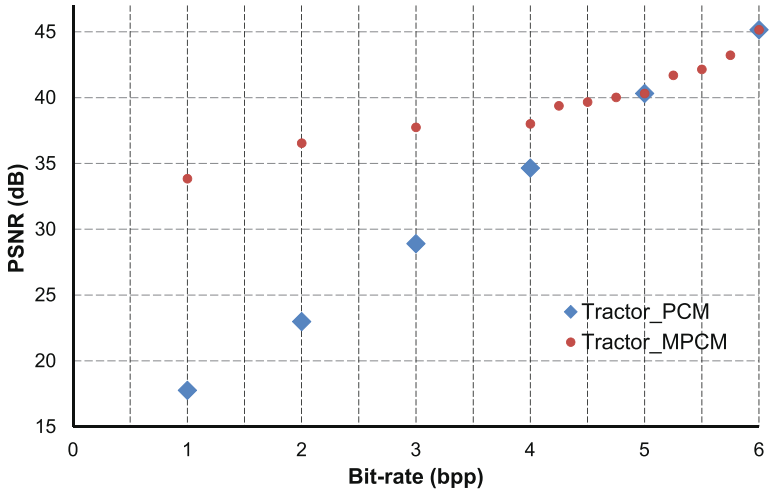


Figure 7 PSNR as a function of the rate of image tractor coded using MPCM and only PCM.

a PCM coding. One advantage of the proposed MPCM algorithm versus PCM is the possibility of compressing intermediate bit rates (nonintegers) due to the different numbers of bits removed in a set of pixels ($l_0 \neq l_1$). In addition, MPCM algorithm overcomes in quality to PCM at high levels of compression . An example of comparison between the MPCM coding and PCM coding is shown in Figure 7 (tractor image), which shows the PSNR values as function of the rate for the image full-HD (1080p). As it can be seen, MPCM obtains the same quality than PCM

Figure 8 A set of four monochromatic images (Zelda 512×512) decoded with the following bit rates. (a) Original image. **(b)** At 6 bpp. **(c)** At 5 bpp. **(d)** At 4 bpp.

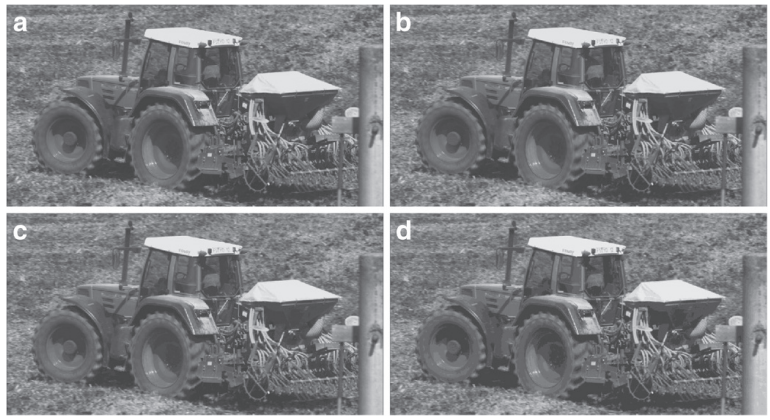


Figure 9 A set of four monochromatic images (Tractor $1,920 \times 1,080$) decoded with the following bit rates. (a) Original image. **(b)** At 6 bpp. **(c)** At 5 bpp. **(d)** At 4 bpp.

at low compression rates. However, at high compression rates, MPCM obtains a PSNR improvement up to 15 dB when compared to PCM.

Furthermore, two images compressed at different bit rates by the algorithm MPCM proposed are shown in Figures 8 and 9. As you can see from the pictures, at 6 and 5 bpp, non-perceptual inequality is observed regarding the original image. However, some differences begin to be appreciated at a rate of 4 bpp, for example, in the set of images of the tractor, one could see a slight distortion inside the rear wheel at 4 bpp.

4.1 Encoder evaluation

Regarding coding/decoding delay, the proposed encoder architecture works at a maximum clock frequency of 204.96 MHz, that is 4.879 ns. Furthermore, the algorithm requires 16,387 cycles to perform the encoding process for an image resolution of 512×512 pixels. Therefore, we require 79.952 μs to encode any image for the aforementioned resolution, which is 12 times faster than the sequential algorithm on an Intel Core 2 CPU at 1.8 GHz with 5 GBytes RAM. As the encoding process does not depend on the internal characteristics of the image, but

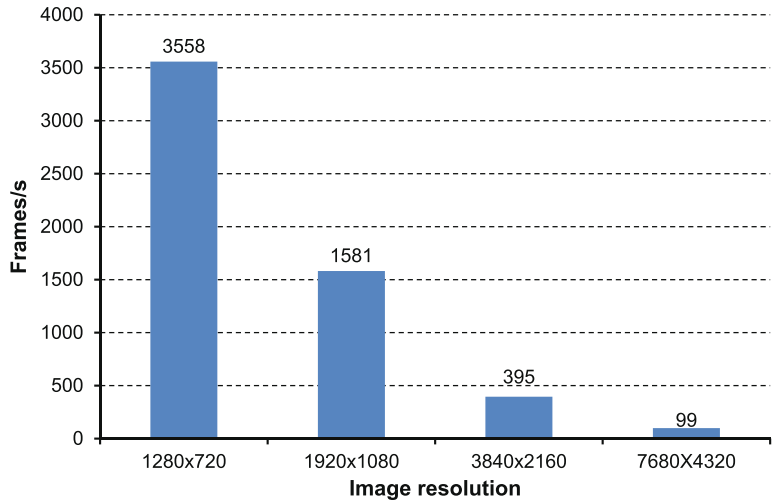


Figure 10 Maximum encoded frames per second for different monochromatic image resolutions.

Table 2 Area used in the FPGA encoder implementation

	Used	Available	Utilization (%)
Number of slices	14	13,300	1
Number of slice registers (as flip-flops)	17	106,400	1
Number of slice LUTs	35	53,200	1
Number of RAMB36	52	140	37
FMax (MHz)	204.96	-	-
Power consumption (mW)	305	-	-

only on the image resolution, in Figure 10, the maximum frame rate achievable for the proposed architecture is presented. As shown, the hardware implemen tation of the MPCM encoder is able to compress up to 3,558 fps for HD-ready resolution (720p) or up to 1,581 fps for full-HD resolution (1080p).

The high-speed encoding process makes high-speed cameras able to capture continuously and grab without the restrictions of the internal RAM size. For example, the proposed MPCM hardware implementation could compress at 4 bpp rate with a reasonable quality and a throughput bandwidth of 1,640 MBytes/s which will extend the capturing time over the internal camera RAM module up to two times or will permit its transmission over a 40-Gbit Ethernet point-to-point access.

The basic elements of a FPGA are configurable logic blocks (CLBs). CLBs architecture includes 6-input look-up tables (LUTs), memory capability within the LUT and register, and shift register functionality. The LUTs in the Zynq-7000 AP SoC can be configured as either one 6-input LUT (64-bit ROMs) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs but common addresses or logic inputs. Each LUT output can optionally be registered in a flip-flop. Four of such LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a configurable logic block (CLB). Four of the eight flip-flops per slice (one flip-flop per LUT) can optionally be configured as latches. Between 25% and 50% of all slices can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers [20].

Table 2 presents the results of the encoder implementation in terms of hardware resources used, indicating the number of used slices, flip-flops, LUTs and 36-KB block RAMs. In addition, it shows an estimation of the power consumed using XPower of Xilinx ISE 14.3, being only 305 mW, due to the high segmentation in the encoder design. As shown, only 1% of all the available area in the FPGA is used, so given the large amount of unused area on the FPGA, we could use it to deploy multiple identical encoders that could run concurrently. Thus, different frames could be encoded simultaneously so as to increment the available recording time of a high-speed camera. To take advantage of this, we would only have to consider an external memory to support the storage of several frames, considering the blocks RAMs used as intermediate buffers. Another option would be to divide the images or frames in different collections of lines which could be encoded in parallel.

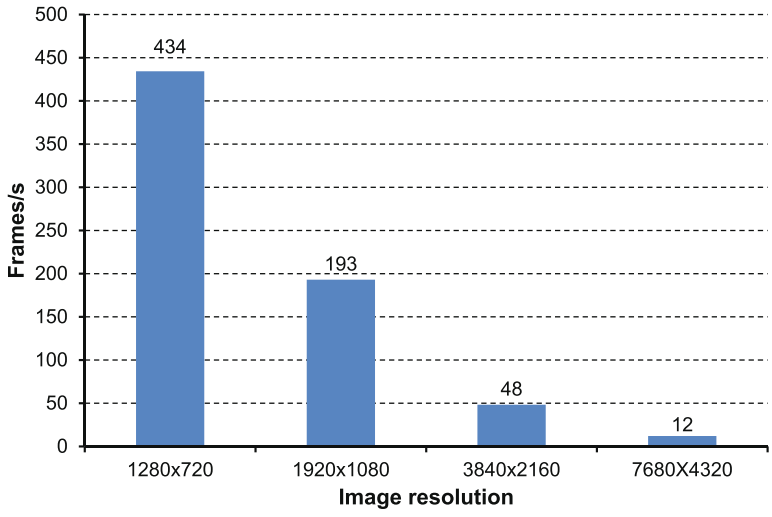


Figure 11 Maximum decoder frames per second for different image resolutions.

Table 3 Area used in the FPGA decoder implementation for parameters $l_0 = 2$, $l_1 = 1$, and $m_1 = 1$

	Used	Available	Utilization (%)
Number of slices	129	13,300	1
Number of slice registers (as flip-flops)	415	106,400	1
Number of slice LUTs	208	53,200	1
Number of RAMB18	5	140	4
FMax (MHz)	154.5	-	-
Power consumption (mW)	221	-	-

In this way, a speed-up of 8× fps would be achieved and as a result, the MPCM encoder would be able to compress up to 12,648 fps for full-HD (1080p) monochromatic resolution.

4.2 Decoder evaluation

As far as the decoder is concerned, the maximum clock frequency obtained for the decoder is 160 MHz with a latency of only 713 cycles. However, the maximum clock frequency has been set at 100 MHz. This frequency is taken as a compromise due to the use of other frequency four times higher provided by the PLL module, as discussed in Section 3.2, since there is a limited frequency for the FPGA used. So, the MPCM decoder is able to recover 400 Mpixels per second at that frequency. On the other hand, the algorithm requires 66,240 cycles to perform the decoding process for an image resolution of 512 × 512 pixels, so 662 μs are needed to decode any image for that resolution, being 70 times faster than the sequential decoding algorithm on an Intel Core 2 CPU at 1.8 GHz with 5 GBytes RAM.

Figure 11 shows the maximum decoding frame rate achievable for the proposed architecture. As shown, the hardware implementation of the MPCM decoder is able to recover up to 434 fps for HD-ready (720p) resolution or up to 193 fps for full-HD (1080p) resolution, which corresponds to a throughput of 50 MBytes/s, making available to reproduce high-definition cinema at high frame rates.

Regarding the occupied board area, Table 3 shows a summary of the hardware resources required by the decoder, which, in a similar way with the encoder, is less than a 1%. The occupied board area could vary depending on the l_0, l_1, m_1 parameters, but in any case it will be lower than 1%. As indicated in Section 3.2, the buffers used have been modeled on single dual-port 18-Kb block RAMs so as to take advantage of the lower consumption compared to distributed memories, besides being faster. Note that the maximum frequency shown in Table 3 is 154.5 MHz, but in our design, we have set this frequency to 100 MHz as explained previously.

5 Conclusions

In this paper, we have presented an efficient FPGA implementation of the MPCM codec. We have shown the quality of the reconstructed frames in terms of PSNR at different compression rates and for several frames with different textures. Regarding coding speed, the results show that our proposed implementation is able to compress a full-HD (1080p) resolution picture at 1,581 fps. The maximum achievable throughput bandwidth of our proposed implementation is 409.84 MBytes/s which permits the continuous grabbing of a nowadays high-speed camera at an image resolution of HD-ready ($1,280 \times 720$p) and a reasonable good quality. But, if the final application requires a higher image quality, our encoder is able to give up to 1,640 MBytes/s at a 2:1 compression rate, incrementing the capturing time over the high-speed camera internal RAM memory. The occupied area of the FPGA used is less than 1% of the total available area, which give us the possibility to replicate several times the encoding system and thus, several frames or different collections of lines of the same image can be compressed in a parallel way.

We have also developed in hardware a MPCM decoder module. Our proposed decoder design is able to recover images at 193 fps for full-HD resolution, with an occupied board area of less than 1%.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
This research was supported by the Spanish Ministry of Education and Science under grant TIN2011-27543-C03-03.

Author details
[1]Physics and Computer Architecture Department, Miguel Hernández University, Elche 03202, Spain. [2]Communications Engineering Department, Miguel Hernández University, Elche 03202, Spain.

Received: 31 January 2013 Accepted: 22 July 2013
Published: 27 August 2013

References

1. Vision Research PHANTOM v641. http://www.visionresearch.com/Products/High-Speed-Cameras/v641. Accessed 7 January 2013.
2. Fastec Imaging TS3 100-S. http://www.fastecimaging.com/products/high-speed-cameras/handheld-cameras/ts3-100-s. Accessed 8 January 2013.
3. PHOTRON FASTCAM SA-X. http://www.photron.com/index.php. Accessed 24 January 2013.
4. i-SPEED 3. http://www.olympus-ims.com/es/ispeed-3/. Accessed 7 January 2013.
5. NS Jayant, P Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. (Prentice-Hall, Englewood Cliffs, 1984)
6. P Gemeiner, W Ponweiser, P Einramhof, M Vincze. Real-time slam with high-speed CMOS camera, in *Proceedings of the 14th International Conference on Image Analysis and Processing* (IEEE Computer Society Washington, 2007), pp. 297–302
7. B Vanhoof, M Peon, MG Lafruit. A scalable architecture for MPEG-4 embedded zero tree coding, in *IEEE Conference on Custom Integrated Circuits* (Town and Country Hotel San Diego, May 1999), pp. 16–19

8. O Ismailoglu, I Benderli, M Korkmaz, T Durna, Y Kolak, A Tekmen. A real time image processing subsystem: Gezgin, in *Proceedings of 16th Annual/USU Conference on Small Satellites,* (Logan, Utah, 12–15 August 2002)
9. LH Chen, WL Liu, OTC Chen, RL Ma. A reconfigurable digital signal processor architecture for high-efficiency MPEG-4 video encoding, in *IEEE International Conference on Multimedia and Expo* (Swiss Federal Institute of Technology Lausanne, 26–29 August 2002)
10. J Ritter, G Fey, P Molitor. Spiht implemented in a XC4000 device, in *Proceedings of IEEE 45th Midwest Symposiumon Circuits and Systems,* (Tulsa, Oklahoma, 4–7 August 2002). vol. 1, pp. 239–242
11. I Urriza, JI Artigas, JI Garcia, LA Barragan, D Navarro. VLSI architecture for lossless compression of medical images using discrete wavelet transform, in *Proceedings of Conference on Design Automation and Test in Europe,* (Belfast, 27–29 August 1998), pp. 196–201
12. J Ahmad, M Ebrahim, FPGA based implementation of baseline JPEG decoder. Int. J. Electrical Comput. Sci. **9**(9), 371–377 (2009)
13. J Rosenthal, JPEG image compression using an FPGA. (PhD Thesis, University of California, 2006)
14. A Descampe, F Devaux, G Rouvroy, B Macq, JD Legat, An efficient FPGA implementation of a flexible JPEG2000 decoder for digital cinema. (PhD Thesis, Université Catholique de Louvain, 2002)
15. X Chen, L Zeng, Q Zhang, W Shi, A novel parallel JPEG compression system based on FPGA. J. Comput. Inf. Syst. **7**(3), 697–706 (2011)
16. Y Wang, S Chen, A Bermak. FPGA implementation of image compression using DPCM and FBAR, in *Proceedings of IEEE International Symposiumon Integrated Circuits, ISIC '07,* (Singapore, 26–28 September 2007), pp. 329–332
17. D Martinez, MM Van Hulle, Generalized boundary adaptation rule for minimizing rth power law distortion in high resolution quantization. Neural Netw. **8**(6), 891–900 (1995)
18. J Prades-Nebot, A Roca, E Delp. Modulo-PCM based encoding for high speed video cameras, in *Proceedings of the 15th IEEE International Conference on Image Processing, ICIP 2008,* (San Diego, CA, 12–15 October 2008), pp. 153–156
19. M Morbee, Optimized information processing in resource-constrained vision systems. (PhD Thesis, Universidad Politécnica de Valencia, Universiteit Gent, 2011)
20. Xilinx, Inc., Zynq-7000 all programmable SoC overview, advance product specification - ds190 (v1.2) (Xilinx, Inc., 2012). http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf. Accessed 20 January 2013.

doi:10.1186/1687-6180-2013-142
Cite this article as: Alcocer *et al.*: MPCM: a hardware coder for super slow motion video sequences. *EURASIP Journal on Advances in Signal Processing* 2013 **2013**:142.