Embedded Systems for Portable and Mobile Video Platforms
Embedded Systems for Portable and Mobile Video Platforms

Guest Editors: Leonel Sousa, Noel O’Connor, Marco Mattavelli, and Antonio Nunez
Editor-in-Chief
Zoran Salcic, University of Auckland, New Zealand

Associate Editors
Sandro Bartolini, Italy
Neil Bergmann, Australia
Shuvra Bhattacharyya, USA
Ed Brinksma, The Netherlands
Paul Caspi, France
Liang-Gee Chen, Taiwan
Dietmar Dietrich, Austria
Stephen A. Edwards, USA
Alain Girault, France
Rajesh K Gupta, USA
Susumu Horiguchi, Japan

Thomas Kaiser, Germany
Bart Kienhuis, The Netherlands
Chong-Min Kyung, Korea
Miriam Leeser, USA
John McAllister, UK
Koji Nakano, Japan
Antonio Nunez, Spain
Sri Parameswaran, Australia
Zebo Peng, Sweden
Marco Platzner, Germany
Marc Pouzet, France

S. Ramesh, India
Partha Roop, New Zealand
Markus Rupp, Austria
Asim Smailagic, USA
Leonel Sousa, Portugal
Jarmo Henrik Takala, Finland
Jean-Pierre Talpin, France
Jürgen Teich, Germany
Dongsheng Wang, China
Contents

Embedded Systems for Portable and Mobile Video Platforms, Leonel A. Sousa, Noel E. O’Connor, Marco Mattavelli, and Antonio Nunez
Volume 2007, Article ID 63250, 2 pages

Low-Complexity Multiple Description Coding of Video Based on 3D Block Transforms, Andrey Norkin, Atanas Gotchev, Karen Egiazarian, and Jaakko Astola
Volume 2007, Article ID 38631, 11 pages

Energy-Efficient Acceleration of MPEG-4 Compression Tools, Andrew Kinane, Daniel Larkin, and Noel O’Connor
Volume 2007, Article ID 28735, 18 pages

Adaptive Motion Estimation Processor for Autonomous Video Devices, T. Dias, S. Momcilovic, N. Roma, and L. Sousa
Volume 2007, Article ID 57234, 10 pages

A Systematic Approach to Design Low-Power Video Codec Cores, Kristof Denolf, Adrian Chirila-Rus, Paul Schumacher, Robert Turney, Kees Vissers, Diederik Verkest, and Henk Corporaal
Volume 2007, Article ID 64569, 14 pages

Thermal-Aware Scheduling for Future Chip Multiprocessors, Kyriakos Stavrou and Pedro Trancoso
Volume 2007, Article ID 48926, 15 pages
Video processing and coding systems are assuming an increasingly important role in a wide range of applications. These include personal communications, wireless multimedia sensing, remote video surveillance, and emergency systems, to name but a few. In such a diverse set of application scenarios, there is a real need to adapt the video processing in general, and video encoding/decoding in particular, to the restrictions imposed by both the envisaged applications and the terminal devices. This is particularly true for portable and battery-supplied devices, in which low-power considerations represent significant challenges to real deployment. The development of novel power-efficient encoding algorithms and architectures suitable for such devices is fundamental to enable the widespread deployment of next generation multimedia applications and wireless network services.

In fact, state-of-the-art implementations of handheld devices for networked electronic media are just one perspective on the actual real challenges posed by the growing ubiquity of video processing and coding on mobile devices. Significant challenges also exist in mapping processing systems developed for fading, noisy, and multipath band-limited transmission channels onto these same devices. Similarly, the requirements for scalable coding associated with networked electronic media also raise issues when handheld mobile devices are considered. A clear need therefore exists to extend, modify, and even create new algorithms, design techniques, and tools targeting architectures and technology platforms as well as addressing scalability, computational load, and energy-efficiency considerations.

The challenge of providing solutions to the requirements of the envisaged application scenarios in terms of image quality and bandwidth is well addressed by new video compression standards, such as the AVC/H.264 joint ITU-ISO/MPEG standard or the upcoming SVC standard. Unfortunately, such high performance is achieved at the expense of an even higher increase in codec complexity. To address all these challenges outlined above, all elements of the solutions have to be addressed, from the encoding algorithms themselves, seeking the best performance-complexity tradeoffs, right down to the design of all architectural elements that need to be conceived and developed with power-efficiency criteria during the design phase. Considering these challenges, this special issue targets to illuminate some important ongoing research in the design and development of embedded systems for portable and mobile video platforms.

For the special issue, we received 13 submissions covering very different areas of expertise within this broad research agenda. After an extremely rigorous review process, only 5 were finally accepted for publication. These 5 papers focused on efficient video coding methods, power-efficient algorithms and architectures for motion estimation and discrete transforms, tools for automatically generating RTL descriptions of video cores, and thermal-aware scheduler algorithms for future on-chip multicore processors. Collectively, we strongly believe that without the pretension of being exhaustive, they represent a “snapshot” of the current state of the art in the area in that they constitute a representative selection of ongoing research.

In a paper entitled “Low-complexity multiple description coding of video based on 3D block transforms”, Andrey Norkin et al. present a multiple description video compression scheme based on three-dimensional transforms, where two balanced descriptions are created from a video sequence. The proposed coder exhibits low computational complexity.
and improved transmission robustness over unreliable networks.

In paper “Energy-efficient acceleration of MPEG-4 compression tools”, Andrew Kinane et al. present some novel hardware accelerator architectures for the most computationally demanding algorithms of MPEG-4 encoding, namely motion estimation and the forward/inverse discrete-cosine transforms, integrating shape-adaptive modes in each of these cases. These accelerators have been designed using general low-energy design approaches both at the algorithmic and architectural levels.

An application-specific instruction set processor (ASIP) to implement data-adaptive motion estimation algorithms is presented by Tiago Dias et al. in a paper entitled “AMEP: adaptive motion estimation processor for autonomous video devices”. This processor is characterized by a specialized data-path and a minimum and optimized instruction set, and is able to adapt its operation to the available energy level in runtime, and is thus a suitable framework in which to develop motion estimators for portable, mobile, and battery-supplied devices.

Kristof Denolf et al. consider the design methodology itself, and in their paper entitled “A systematic approach to design of low power video codec cores”, describe how a memory and communication-centric design methodology can be targeted to the development of dedicated cores for embedded systems. This methodology is adopted to design an MPEG-4 simple profile video codec using both FPGA and ASIC technologies.

K. Stavrou and P. Trancoso take a different perspective and analyze the evolution of thermal issues for future chip multiprocessor architectures in a paper entitled “thermal-aware scheduling for future chip multiprocessors”. They show that as the number of on-chip cores increases, the thermal-induced problems will worsen. In order to minimize or even eliminate these problems, thermal-aware scheduler algorithms are proposed and their relative efficiency is quantified.

In conclusion, we hope that you will enjoy this special issue and the range of topics covered in this important area.

ACKNOWLEDGMENTS

We would like to express our gratitude to all authors for the high quality of their submissions. In addition, we would like to thank all the reviewers for their rigorous, constructive, and timely reviews that enabled us to put together this special issue.

Leonel A. Sousa
Noel E. O’Connor
Marco Mattavelli
Antonio Nunez
Research Article

Low-Complexity Multiple Description Coding of Video Based on 3D Block Transforms

Andrey Norkin, Atanas Gotchev, Karen Egiazarian, and Jaakko Astola

Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland

Received 28 July 2006; Revised 10 January 2007; Accepted 16 January 2007

The paper presents a multiple description (MD) video coder based on three-dimensional (3D) transforms. Two balanced descriptions are created from a video sequence. In the encoder, video sequence is represented in a form of coarse sequence approximation (shaper) included in both descriptions and residual sequence (details) which is split between two descriptions. The shaper is obtained by block-wise pruned 3D-DCT. The residual sequence is coded by 3D-DCT or hybrid, LOT+DCT, 3D-transform. The coding scheme is targeted to mobile devices. It has low computational complexity and improved robustness of transmission over unreliable networks. The coder is able to work at very low redundancies. The coder is able to work at very low redundancies. The coding scheme is simple, yet it outperforms some MD coders based on motion-compensated prediction, especially in the low-redundancy region. The margin is up to 3 dB for reconstruction from one description.

Copyright © 2007 Andrey Norkin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Nowadays, video is more often being encoded in mobile devices and transmitted over less reliable wireless channels. Traditionally, the objective in video coding has been to achieve high compression, which was attained with the cost of increasing encoding complexity. However, portable devices, such as camera phones, still lack enough computational power and are energy-consumption constrained. Besides, a highly compressed video sequence is more vulnerable to transmission errors, which are often present in wireless networks due to multipath fading, shadowing, and environmental noise. Thus, there is a need of a low-complexity video coder with acceptable compression efficiency and strong error-resilience capabilities.

Lower computational complexity in transform-based video coders can be achieved by properly addressing the motion estimation problem, as it is the most complex part of such coders. For the case of high and moderate frame rates ensuring smooth motion, motion-compensated (MC) prediction can be replaced by a proper transform along the temporal axis to handle the temporal correlation between frames in the video sequence. Thus, the decorrelating transform adds one more dimension, becoming a 3D one, and if a low complexity algorithm for such a transform exists, savings in overall complexity and power consumption can be expected compared to traditional video coders [1–4]. Discrete cosine transform (DCT) has been favored for its very efficient 1D implementations. As DCT is a separable transform, efficient implementations of 3D-DCT can be achieved too [2, 3, 5]. Previous research on this topic shows that simple (baseline) 3D-DCT video encoder is three to four times faster than the optimized H.263 encoder [6], for the price of some compression efficiency loss, quite acceptable for portable devices [7].

A 3D-DCT video coder is also advantageous in terms of error resilience. In MC-based coders, the decoding error would propagate further into subsequent frames until the error is corrected by an intracoded frame. The error could also spread over the bigger frame area because of motion-compensated prediction. Unlike MC-based coders, 3D-DCT video coders enjoy no error propagation in the subsequent frames. Therefore, we have chosen the 3D-DCT video coding approach for designing a low-complexity video coder with strong error resilience.

A well-known approach addressing the source-channel robustness problem is so-called multiple description coding (MDC) [8]. Multiple encoded bitstreams, called descriptions, are generated from the source information. They are correlated and have similar importance. The descriptions are independently decodable at the basic quality level and, when several descriptions are reconstructed together, improved
quality is obtained. The advantages of MDC are strengthened when MDC is connected with multipath (multichannel) transport [9]. In this case, each bitstream (description) is sent to the receiver over a separate independent path (channel), which increases the probability of receiving at least one description.

Recently, a great number of multiple description (MD) video coders have appeared, most of them based on MC prediction. However, MC-based MD video coders risk having a mismatch between the prediction loops in the encoder and decoder when one description is lost. The mismatch could propagate further in the consequent frames if not corrected. In order to prevent this problem, three separate prediction loops are used at the encoder [10] to control the mismatch. Another solution is to use a separate prediction loop for every description [11, 12]. However, both approaches decrease the compression efficiency and the approach in [10] also leads to increased computational complexity and possibly to increased power consumption. A good review of MDC approaches to video coding is given in [13]. A number of MD and error-resilient video coders based on 3D transforms (e.g., wavelets, lapped orthogonal transforms (LOT), DCT) have been proposed [14–17].

In this work, we investigate a two-stage multiple description coder based on 3D transforms, denoted by 3D-MDC. This coder does not exploit motion compensation as initially proposed in [18]. Using 3D transform instead of motion compensated prediction reduces the computational complexity of the coder, meanwhile eliminating the problem of mismatch between the encoder and decoder. The proposed MD video coder is a generalization of our 2-stage image MD coding approach [19] to coding of video sequences [18]. Designing the coder, we are targeting balanced computational load between the encoder and decoder. The coder should be able to work at a very low redundancy introduced by MD coding and be competitive with MD video coders based on motion-compensated prediction.

The paper is organized as follows. Section 2 overviews the encoding and decoding processes in general while Section 3 describes each block of the proposed scheme in detail. Section 4 presents the analysis of the proposed scheme and Section 5 discusses its computational complexity. Section 6 offers a packetization strategy; Section 7 presents the simulation results; while Section 8 concludes the paper.

2. GENERAL CODING SCHEME

2.1. Encoder operation

In our scheme, a video sequence is coded in two stages as shown in Figure 1. In the first stage (dashed rectangle), a coarse sequence approximation, called shaper, is obtained and included in both descriptions. The second stage produces enhancement information, which has higher bitrate and is split between two descriptions. The idea of the method is to get a coarse signal approximation which is the best possible for the given bitrate while decorrelating the residual sequence as much as possible.

The operation of the proposed encoder is described in the following. First, a sequence of frames is split into groups of 16 frames. Each group is split into 3D cubes of size $16 \times 16 \times 16$. 3D-DCT is applied to each cube. The lower-frequency DCT coefficients in the $8 \times 8 \times 8$ cube are coarsely quantized with quantization step $Q_s$ and entropy-coded (see Figure 2(a)) composing the shaper, other coefficients are set to zero. Inverse quantization is applied to these coefficients followed by the inverse 3D-DCT. An optional deblocking filter serves to remove the block edges in spatial domain. Then, the sequence reconstructed from the shaper is subtracted from the original sequence to get the residual sequence.

The residual sequence is coded by a 3D block transform and transform coefficients are finely quantized with a uniform quantization step ($Q_r$), split into two parts in a manner shown in Figure 2(b), and entropy-coded. One part together with the shaper forms Description 1, while the second part combined again with the shaper forms Description 2. Thus, each description consists of the shaper and half of the transform volumes of the residual sequence.
The shaper is included in both descriptions to facilitate successful reconstruction when one description is lost. Thus, the redundancy of the proposed coder is only determined by the shaper quality, which is controlled by the shaper quantization step $Q_s$. A larger quantization step corresponds to lower level of redundancy and lower quality of side reconstruction (reconstruction from only one description). Alternatively, a smaller quantization step results in higher-quality side reconstruction. The quality of the two-channel reconstruction is controlled by the quantization step $Q$, used in the coding of the residual sequence. As the residual volumes are divided into two equal parts, the encoder produces balanced descriptions both in terms of PSNR and bitrate.

### 2.2. Decoder operation

The decoder (see Figure 3) operates as follows. When the decoder receives two descriptions, it extracts the shaper ($X_s$) from one of the descriptions. Then, the shaper is entropy-decoded and inverse quantization is applied. The $8 \times 8 \times 8$ volume of coefficients is zero-padded to the size $16 \times 16 \times 16$, and inverse DCT is applied. The deblocking filter is applied if it was applied in the encoder.

In case of central reconstruction (reconstruction from two descriptions), each part of the residual sequence ($X_1$ and $X_2$) is extracted from the corresponding description and entropy decoded. Then, volumes of the corresponding descriptions are decoded and combined together as in Figure 2(b). The inverse quantization and inverse transform (IDCT or Hybrid inverse transform) are applied to coefficients and the residual sequence is added to the shaper to obtain the reconstruction of the original sequence.

We term the reconstruction from one description, for example, Description 1, as side reconstruction (reconstruction from Description 2 is symmetrical). The side decoder scheme can be obtained from Figure 3 if the content of the dashed rectangle is removed. In this case, the shaper is reconstructed from its available copy in Description 1. The residual sequence, however, has only half of the coefficient volumes ($X_1$). The missing volumes $X_2$ are simply filled with zeros. After that, the decoding process is identical to that of the central reconstruction. As the residual sequence has only half of the coefficient volumes, the side reconstruction has lower, however, still acceptable quality. For example, sequence “silent voice” coded at 64.5 kbps with 10% redundancy can be reconstructed with $\text{PSNR} = 31.49 \text{ dB}$ from two descriptions, and $26.91 \text{ dB}$ from one description (see Table 2).

## 3. Detailed System Description

### 3.1. The coarse sequence approximation

The idea of the first coding stage is to concentrate as much information as possible into the shaper within strict bitrate constraints. We would also like to reduce artifacts and distortions appearing in the reconstructed coarse approximation. The idea is to reduce spatial and temporal resolutions of the coarse sequence approximation in order to code it more efficiently with lower bitrate [20]. Then, the original resolution sequence can be reconstructed by interpolation as a post-processing step. The original resolution sequence can be reconstructed by interpolation as a post-processing step. A good interpolation and decimation method would concentrate more information in the coarse approximation and correspondingly make the residual signal closer to white noise. A computationally inexpensive approach is to embed interpolation in the 3D transform.

The downscaling factor for the shaper was chosen equal to two in both spatial and temporal directions. The proposed scheme is able to use other downscaling factors equal to powers of two. However, the downscaling factor two has been chosen as the one producing the best results for QCIF and CIF resolutions. To reduce computational complexity, we combine downsampling with forward transform (and backward transform with interpolation). Thus, the original sequence is split into volumes of size $16 \times 16 \times 16$, and 3D-DCT is applied to each volume. Pruned DCT is used in this stage that allows to reduce computational complexity (see Figure 2(a)). The transform size of $16 \times 16 \times 16$ has been chosen as a compromise between the compression efficiency and computational complexity.

Only $8 \times 8 \times 8$ cubes of low-frequency coefficients in each $16 \times 16 \times 16$ coefficient volume are used; other coefficients are set to zero (see Figure 2(a)). The AC coefficients of the $8 \times 8 \times 8$ cube are uniformly quantized with quantization step $Q_s$. DC coefficients are quantized with the quantization step $Q_{DCT}$.

In the $8 \times 8 \times 8$ volume, we use coefficient scanning described in [21], which is similar to a 2D zigzag scan. Although there exist more advanced types of quantization and scanning of 3D volumes [1, 22], we have found that simple scanning performs quite well. An optional deblocking filter may be used to eliminate the blocking artifacts caused by quantization and coefficient thresholding.

The DC coefficients of the transformed shaper volumes are coded by DPCM prediction. The DC coefficient of the volume is predicted from the DC coefficient of the temporally preceding volume. As the shaper is included in both descriptions, there is no mismatch between the states of the encoder and decoder when one description is lost.
First, the DC coefficient prediction errors and the AC coefficients undergo zero run-length (RL) encoding. It combines runs of successive zeros and the following nonzero coefficients into two-tuples where the first number is the number of leading zeros, and the second number is the absolute value of the first nonzero coefficient following the zero-run.

Variable-length encoding is implemented as a standard Huffman encoder similar to the one in H.263 [6]. The codebook has the size 100 and is calculated for the two tuples which are the output of RL-coding. All values exceeding the range of the codebook are encoded with an “escape” code followed by the actual value. Two different codebooks are used: one for coding the shaper and another for coding the residual sequence.

3.2. Residual sequence coding

The residual sequence is obtained by subtracting the reconstructed shaper from the original sequence. As the residual sequence consists of high-frequency details, we do not add any redundancy at this stage. The residual sequence is split into groups of 8 frames in such a way that two groups of 8 frames correspond to one group of 16 frames obtained from the coarse sequence approximation. Each group of 8 frames undergoes block 3D transform. The transform coefficients are uniformly quantized with the quantization step $Q_r$ and split between two descriptions in a pattern shown in Figure 2(b).

Two different transforms are used in this work to code the residual sequence. The first transform is 3D-DCT and the second is a hybrid transform. The latter consists of the lapped orthogonal transform (LOT) [23] in vertical and horizontal directions, and DCT in temporal direction. Both DCT and the hybrid transform produce $8 \times 8 \times 8$ volumes of coefficients, which are split between the two descriptions. Using LOT in spatial domain smooths blocking artifacts when reconstructing from one description. In this case, LOT spatially spreads the error caused by loosing transform coefficient blocks. Although LOT could be applied in the temporal direction to reduce blocking artifacts in temporal domain too, we avoid using it because of additional delay it introduces in the encoding and decoding processes.

As will be demonstrated in Section 7, the hybrid transform outperforms DCT in terms of PSNR and visual quality. Moreover, using LOT in spatial dimensions gives better visual results compared to DCT. However, blocking artifacts introduced by coarse coding of the shaper are not completely concealed by the residual sequence coded with the hybrid transform. These artifacts impede efficient compression of the residual sequence by the hybrid transform. Therefore, the deblocking filter is applied to the reconstructed shaper (see Figure 1) prior to subtracting it from the original sequence. In the experiments, we use the deblocking filter from H.263+ standard [6].

In the residual sequence coding, the transform coefficients are uniformly quantized with the quantization step $Q_r$. DC prediction is not used in the second stage to avoid the mismatch between the states of the encoder and decoder if one description is lost. The scanning of coefficients is 3D-zigzag scanning [21]. The entropy coding is RL coding followed by Huffman coding with a codebook different from the one used in coding the coarse sequence approximation.

4. SCHEME ANALYSIS

4.1. Redundancy and reconstruction quality

Denote by $D_0$ the central distortion (distortion when reconstructing from two descriptions), and by $D_1$ and $D_2$ the side distortions (distortions when reconstructing from only one description). In case of balanced descriptions, $D_1 = D_2$. Denote as $D_s$ the distortion of the video sequence reconstructed only from the shaper. Consider 3D-DCT coding of the residual sequence. The side distortion $D_1$ is formed by the blocks, half of which are coded with the distortion $D_0$, and half with the shaper distortion $D_s$. Here we assume that all blocks of Description 1 have the same expected distortion as blocks of Description 2. Consequently,

$$D_1 = \frac{1}{2} (D_s + D_0). \quad (1)$$

Expression (1) can also be used in case the hybrid transform is used for coding the residual. As LOT is by definition an orthogonal transform, mean-squared error distortion in spatial domain is equal to the distortion in the transform domain.
The side distortion in the transform domain is determined by loosing half of the transform coefficient blocks. Thus, expression (1) is also valid for hybrid transform. It is obvious that $D_1$ depends on the bitrate $R_s$ allocated to the shaper. Then, we can write (1) as

$$D_1(R_s, R_r) = \frac{1}{2}(D_3(R_s) + D_0(R_s, R_r)),$$  \hspace{1cm} (2)

where $R_r$ is the bitrate allocated for coding the residual sequence and $R_s$ is the bitrate allocated to the shaper. For higher bitrates, $D_1(R_s) \gg D_0(R_s)$, and $D_1$ mostly depends on $R_s$.

The redundancy $\rho$ of the proposed scheme is the bitrate allocated to the shaper, $\rho = R_s$. The shaper bitrate $R_s$ and the side reconstruction distortion $D_1$ depend on the quantization step $Q_s$ and the characteristics of the video sequence. The central reconstruction distortion $D_0$ is mostly determined by the quantization step $Q_r$.

Thus, the encoder has two control parameters: $Q_s$ and $Q_r$. By changing $Q_r$, the encoder controls the central distortion. By changing $Q_s$, the encoder controls the redundancy and the side distortion.

### 4.2. Optimization

The proposed scheme can be optimized for changing channel behavior. Denote by $p$ the probability of the packet loss and by $R$ the target bitrate. Then, in case of balanced descriptions we have to minimize

$$2p(1-p)D_1 + (1-p)^2D_0$$  \hspace{1cm} (3)

subject to

$$2R_s + R_r \leq R.$$  \hspace{1cm} (4)

Taking into consideration (1), expression (3) can be transformed to the unconstrained minimization task

$$J(R_s, R_r) = p(1-p)(D_3(R_s) + D_0(R_s, R_r)) + (1-p)^2D_0(R_s, R_r) + \lambda(2R_s + R_r - R).$$  \hspace{1cm} (5)

It is not feasible to find the distortion-rate functions $D_0(R_s, R_r)$ and $D_1(R_s)$ in real-time to solve the optimization task. Instead, the distortion-rate (D-R) function of a 3D coder can be modeled as

$$D(R) = b2^{-aR} - c,$$  \hspace{1cm} (6)

where $a$, $b$, and $c$ are parameters, which depend on the characteristics of the video sequence. Hence,

$$D_0(R_s, R_r) = b2^{-a(R_s+R_r)} - c.$$  \hspace{1cm} (7)

Assuming that the source is successively refinable in regard to the squared-error distortion measure (this is true, e.g., for i.i.d. Gaussian source [24]) we can write

$$D_0(R_s, R_r) = b2^{-a(R_s+R_r)} - c.$$  \hspace{1cm} (8)

Then, substituting (7) and (8) into (5) and differentiating the resulting Lagrangian with respect to $R_s$, $R_r$, and $\lambda$, we can find a closed form solution of the optimization task (5). The obtained optimal values of bitrates $R_s$ and $R_r$ are

$$R_s^* = \frac{1}{2}R + \frac{1}{2a}\log_2(p),$$

$$R_r^* = \frac{1}{a}\log_2(p),$$  \hspace{1cm} (9)

where $R_s^*$ and $R_r^*$ are rates of the shaper and the residual sequence, respectively.

Hence, the optimal redundancy $\rho^*$ of the proposed scheme under above assumptions is

$$\rho^* = R_s^* = \frac{1}{2}R + \frac{1}{2a}\log_2(p).$$  \hspace{1cm} (10)

The optimal redundancy $\rho^*$ depends on the target bitrate $R$, the probability of packet loss $p$, and parameter $a$ of the source D-R function. It does not depend on D-R parameters $b$ and $c$. We have found that parameter $a$ usually takes similar values for video sequences with the same resolution and frame rates. Thus, one does not need to estimate $a$ in real-time. Instead, one can use a typical value of $a$ to perform optimal bit allocation during encoding. For example, sequences with CIF resolution and 30 frames per second usually have the value of $a$ between 34 and 44 for bitrates under 1.4 bits per pixel.

One notices that for values $R$ and $p$ such that $R \leq -(1/a)\log_2(p)$, the optimal redundancy $\rho^*$ is zero or negative. For these values of $R$ and $p$, the encoder should not use MDC. Instead, single description coding should be used. It is seen from (10) that the upper limit for redundancy is $R/2$, which is obtained for $p = 1$. That means that all the bits are allocated to the shaper, which is duplicated in both descriptions.

### 5. COMPUTATIONAL COMPLEXITY

To perform a 3D-DCT of an $N \times N \times N$ cube, one has to perform $3N^2$ one-dimensional DCTs of size $N$. However, if one needs only the $N/2 \times N/2 \times N/2$ low-frequency coefficients, as in the case of the shaper coding, a smaller amount of DCTs need to be computed. Three stages of separable row-column-frame (RCF) transform require $[N^2 + 1/2N^2 + 1/4N^2] = 1.75N^2$ DCTs for one cube. The same is true for the inverse transform.

The encoder needs only the 8 lowest coefficients of 1D-DCT. For this reason, we use pruned DCT as in [25]. The computation of the 8 lowest coefficients of pruned DCT II [26] of size 16 requires 24 multiplications and 61 additions [25]. That gives 2.625 multiplications and 6.672 additions per point and brings substantial reduction in computational complexity. For comparison, full separable DCT II (decimation in frequency (DIF) algorithm) [26] of size 16 would require 6 multiplications and 15.188 additions per point.

The operation count for different 3D-DCT schemes is provided in Table 1. The adopted “pruned” algorithm is compared to fast 3D vector-radix decimation-in-frequency DCT (3D VR DCT) [5] and row-column-frame (RCF) approach, where 1D-DCT is computed by DIF algorithm [26]. One can see that the adopted “pruned” algorithm has the
Table 1: Operations count for 3D-DCT II. Comparison of algorithms.

| Transform | Pruned 16 × 16 × 16 | 3D VR 16 × 16 × 16 | RCF 16 × 16 × 16 | 3D VR 8 × 8 × 8 | RCF 8 × 8 × 8 |
|-----------|---------------------|-------------------|-----------------|-----------------|----------------|
| Mults/point | 2.625               | 3.5              | 6               | 2.625           | 4.5            |
| Adds/point | 6.672               | 15.188           | 15.188          | 10.875          | 10.875         |
| Mults+adds/point | 9.297         | 18.688           | 21.188          | 13.5            | 15.375         |

lowest computational complexity. In terms of operations per pixel, partial DCT 16 × 16 × 16 is less computationally expensive than full 8 × 8 × 8 DCT used to code the residual sequence.

In [7], a baseline 3D-DCT encoder is compared to the optimized H.263 encoder [27]. It was found [7] that baseline 3D-DCT encoder is up to four times faster than the optimized H.263 encoder. In the baseline 3D-DCT encoder [7], DCT was implemented by RCF approach, which gives 15.375 operations/point. In our scheme, forward pruned 3D-DCT for the shaper requires only 9.3 op/point. Adding the inverse transform, one gets 18.6 op/points. The 8 × 8 × 8 DCT of the residual sequence can be implemented by 3D VR DCT [5], which requires 13.5 op/point. Thus, the overall complexity of the transforms used in the proposed encoder is estimated as 32.1 op/point, that is about twice higher than the complexity of the transforms used in baseline 3D-DCT (15.375 op/point).

The overall computational complexity of the encoder includes quantization and entropy coding of the shaper coefficients. However, the number of coefficients coded in the shaper is eight times lower than the number of coefficients in the residual sequence as only 512 lower DCT coefficients in each 16 × 16 × 16 block are coded. Thus, quantization and entropy coding of the shaper would take about 8 times less computations than quantization and entropy coding of the residual sequence. Thus, we estimate that the overall complexity of the proposed encoder is not more than twice the complexity of baseline 3D-DCT [7]. This means that the proposed coder has up to two times lower-computational complexity than the optimized H.263 [27]. The difference in computational complexity between the proposed coder and H.263+ with scalability (providing error resilience) is even bigger. However, the proposed coder has single description performance similar or even higher than H.263+ [6] with SNR scalability, as shown in Section 7.

6. PACKETIZATION AND TRANSMISSION

The bitstream of the proposed video coder is packetized as follows. A group of pictures (16 frames) is split into 3D-volumes of size 16 × 16 × 16. One packet should contain one or more shaper volumes, which gives 512 entropy-coded coefficients (due to thresholding).

In case of single description coding, one shaper volume is followed by eight spatially corresponding volumes of the residual sequence, which have the size of 8 × 8 × 8. In case of multiple description coding, a packet from Description 1 contains a shaper volume and four residual volumes taken in the pattern shown in Figure 2(b). Description 2 contains the same shaper volume and four residual volumes, which are not included into Description 1. If the size of such a block (one shaper volume and four residual volumes) is small, several blocks are packed into one packet.

The proposed coder uses DPCM prediction of DC coefficients in the shaper volumes. The DC coefficient is predicted from the DC coefficient of the temporally preceding volume. If both descriptions containing the same shaper volume are lost, DC coefficient is estimated as the previous DC coefficient in the same spatial location or as an average of DC coefficients of the spatially adjacent volumes. This concealment may introduce mismatch in DPCM loop between the encoder and decoder. However, the mismatch does not spread out of the border of this block. The mismatch is corrected by the DC coefficient update which can be requested over a feedback channel or may be done periodically.

To further improve the robustness against burst errors, the bitstream can be reordered in a way that descriptions corresponding to one 3D volume are transmitted in the packets which are not consecutive. It will decrease the probability that both descriptions are lost due to consequent packet losses. Another solution to improve the error resilience is to send the packets of Description 1 over one link, and packets from Description 2 over another link.

7. SIMULATION RESULTS

This section presents the comparison of the proposed MD coder with other MD coders. The experiments are performed on sequences “Tempe-te” (CIF, 30 fps, 10 s), “silent voice” (QCIF, 15 fps, 10 s), and “Coastguard” (CIF, 30 fps). We measure the reconstruction quality by using the peak signal-to-noise ratio (PSNR). The distortion is average luminance PSNR over time, all color components are coded. We compare our scheme mainly with H.263-based coders as our goal is low-complexity encoding. Apparently, the proposed scheme cannot compete with H.264 in terms of compression performance. However, H.264 encoders are much more complex.

7.1. Single description performance

Figure 4 plots PSNR versus bitrate for the sequence “Tempe-te.” The compared coders are single description coders. “3D-2stage” coder is a single-description variety of the coder described above. The shaper is sent only once, and the residual sequence is sent in a single description. “3D-DCT” is a simple 3D-DCT coder described in [1, 7]. “H.263” is a Telenor implementation of H.263. “H.263-SNR” is an H.263+ with SNR scalability, implemented at the University.
of British Columbia [28, 29]. One can see that H.263 coder outperforms other coders. Our 3D-2stage has approximately the same performance as H.263+ with SNR scalability and its PSNR is half to one dB lower than that of H.263+. Simple 3D-DCT coder showed the worst performance.

Figure 5 shows PSNR of the first 100 frames of “Tempete” sequence. The sequence is encoded to target bitrate 450 kbps. Figure 5 demonstrates that 3D-DCT coding exhibits temporal degradation of quality on the borders of 8-frame blocks. These temporal artifacts are caused by block-wise DCT and perceived like abrupt movements. These artifacts can be efficiently concealed with postprocessing on the decoder side. In this experiment, we applied MPEG-4 deblocking filter [30] to block borders in temporal domain. As a result, temporal artifacts are smoothed. The perceived quality of the video sequence has also improved. Some specialized methods for deblocking in temporal domain can be applied as in [31]. Postprocessing in temporal and spatial domains can also improve reconstruction quality in case of description loss. In the following experiments, we do not use postprocessing in order to have fair comparison with other MDC methods.

### 7.2. Performance of different residual coding methods

In the following, we compare the performance of MD coders in terms of side reconstruction distortion, while they have the same central distortion. Three variants of the proposed 3D-2sMDC coder are compared. These MD coders use different schemes for coding the residual sequence. “Scheme 1” is the 2-stage coder, which uses hybrid transform for the residual sequence coding and the deblocking filtering of the shaper. “Scheme 2” employs DCT for coding the residual sequence. “Scheme 3” is similar to “Scheme 2” except that it uses the deblocking filter (see Figure 1). We have compared these schemes with simple MD coder based on 3D-DCT and MDSQ [32]. MDSQ is applied to the first $N$ coefficients of $8 \times 8 \times 8$ 3D-DCT cubes. Then, MDSQ indices are sent to corresponding descriptions, and the rest of $512 - N$ coefficients are split between two descriptions (even coefficients go to Description 1 and odd coefficients to Description 2).

Figure 6 shows the result of side reconstruction for the reference sequence “Tempete.” The average central distortion (reconstruction from both descriptions) is fixed for all encoders, $D_0 = 28.3$ dB. The mean side distortion (reconstruction from one description) versus bitrate is compared. One can see that “Scheme 1” outperforms other coders, especially in the low-redundancy region. One can also see that the deblocking filtering applied to the shaper (“Scheme 3”) does not give much advantage for the coder using 3D-DCT for coding the residual sequence. However, the deblocking filtering of the shaper is necessary in “Scheme 1” as it considerably enhances visual quality. The deblocking filtering requires twice less operations comparing to the sequence of the same format in H.263+ because the block size in the shaper is twice larger than that in H.263+. All the three variants of our coder outperform the “3D-MDSQ” coder to the extent of 2 dB.

### 7.3. Network performance of the proposed method

Figure 7 shows performance of the proposed coder in network environment with error bursts. In this experiment, bursty packet loss behavior is simulated by a two-state Markov model. These two states are G (good) when packets are correctly received and B (bad) when packets are either lost or delayed. This model is fully described by transition probabilities $p_{BG}$ from state B to state G and $p_{GB}$ from G to B.
The model can also be described by average loss probability 

\[ P_B = \Pr(B) = \frac{p_{GB}}{(p_{GB} + p_{BG})} \]

and the average burst length 

\[ L_B = \frac{1}{P_{BG}}. \]

In the following experiment, the sequence “Tempete” (CIF, 30 fps) has been coded to bitrate 450 kbps into packets not exceeding the size of 1000 bytes for one packet. The coded sequence is transmitted over two channels modeled by two-state Markov models with \( P_B = 0.1 \) and \( L_B = 5 \). Packet losses in Channel 1 are uncorrelated with errors in Channel 2. Packets corresponding to Description 1 are transmitted over Channel 1, and packets corresponding to Description 2 are transmitted over Channel 2. Two channels are used to ensure uncorrelated losses of Description 1 and Description 2. Similar results can be achieved by interleaving packets (descriptions) corresponding to the same spatial locations. When both descriptions are lost, error concealment described in Section 6 is used. Optimal redundancy for “Tempete” sequence estimated by (10) for bitrate 450 kbps (0.148 bpp) is 21%.

Figure 7 shows network performance of 3D-2sMDC and 3D-2sMDC with postprocessing (temporal deblocking). The performance of a single description 3D-2stage coder with postprocessing in a lossless environment is also given in Figure 7 as a reference. One can see that using MDC for error resilience helps to maintain an acceptable level of quality when transmitting over network with packet losses.

### 7.4. Comparison with other MD coders

The next set of experiments is performed on the first 16 frames of the reference sequence “Coastguard” (CIF, 30 fps). The first coder is the proposed 3D-2sMDC coder Scheme 1. The “H.263 spatial” method exploits H.263+ [29] to generate layered bitstream. The base layer is included in both descriptions while the enhancement layer is split between two descriptions on a GOB basis. The “H.263 SNR” is similar to the previous method with the difference that it uses SNR scalability to create two layers.

Figure 8 plots the single description distortion versus bitrate of the “Coastguard” sequence for the three coders described above. The average central distortion is \( D_0 = 28.5 \) dB. One can see that 3D-2stage method outperforms the two other methods.

The results indicate that the proposed MD coder based on 3D transforms outperforms simple MD coders based on H.263+ and the coder based on MDSQ and 3D-DCT. For the coder with SNR scalability, we were not able to get the bitrates as low as we have got with our “3D-2stage” method.

Another set of experiments is performed on the reference sequence “Silent voice” (QCIF, 15 fps). The proposed 3D-2sMDC coder is compared with MDTC coder that uses three prediction loops in the encoder [10, 33]. The 3D-2sMDC coder exploits “Scheme 1” as in the previous set of experiments. The rate-distortion performance of these two coders is shown in Figure 9. The PSNR of two-description reconstruction of 3D-2sMDC coder is \( D_0 = 31.47 - 31.57 \) dB and central distortion of MDTC coder is \( D_0 = 31.49 \) dB.

The results show that the proposed 3D-2sMDC coder outperforms the MDTC coder, especially in a low-redundancy region. The superior side reconstruction performance of our coder could be explained by the following. MC-based multiple description video coder has to control the mismatch between the encoder and decoder. It could be done, for example, by explicitly coding the mismatch signal, as it is done in [10, 33]. In opposite, MD coder based on 3D transforms does not need to code the residual signal, thus,
and the DCT. This property is common for most of 3D transform video coders. We suppose that most of modern mobile devices have enough memory to perform the encoding.

Figure 10 shows frame 13 of the reference sequence Tempete reconstructed from both descriptions (Figure 10(a)) and from Description 1 alone (Figure 10(b)). The sequence is coded by 3D-2sMDC (Scheme 1) encoder to bitrate $R = 880$ kbps. One can see that although the image reconstructed from one description has some distortions caused by loss of transform coefficient volumes of the residual sequence, the overall picture is smooth and pleasant to the eye.

Table 2: Reconstruction results. Sequence “Silent voice.”

| Central PSNR (dB) | Mean side PSNR (dB) | Bitrate (kbps) | Redundancy (%) |
|-------------------|---------------------|----------------|----------------|
| 31.49             | 26.91               | 64.5           | 9.8            |
| 31.51             | 27.34               | 65.5           | 11.4           |
| 31.51             | 27.83               | 66.8           | 13.7           |
| 31.57             | 28.47               | 70.3           | 19.6           |
| 31.52             | 29.05               | 74.2           | 26.3           |
| 31.47             | 29.54               | 81.2           | 38.2           |
| 31.53             | 29.97               | 89.2           | 51.8           |

Figure 8: Sequence “Coastguard,” mean side reconstruction. $D_0 \approx 28.5$ dB.

Figure 9: Sequence “Silent voice,” mean side reconstruction. $D_0 \approx 31.53$ dB.

A drawback of our coder is relatively high delay. High delays are common for coders exploiting 3D transforms (e.g., coders based on 3D-DCT or 3D-wavelets). Waiting for 16 frames to apply 3D transform introduces additional delay of slightly more than half a second for the frame rate 30 fps and about one second for 15 fps. The proposed coder also needs larger memory than MC-based video coder, as it is required to keep the 16 frames in the buffer before applying the DCT. This property is common for most of 3D transform video coders. We suppose that most of modern mobile devices have enough memory to perform the encoding.

Figure 10 shows frame 13 of the reference sequence Tempete reconstructed from both descriptions (Figure 10(a)) and from Description 1 alone (Figure 10(b)). The sequence is coded by 3D-2sMDC (Scheme 1) encoder to bitrate $R = 880$ kbps. One can see that although the image reconstructed from one description has some distortions caused by loss of transform coefficient volumes of the residual sequence, the overall picture is smooth and pleasant to the eye.
8. CONCLUSION

We have proposed an MDC scheme for coding of video which does not use motion-compensated prediction. The coder exploits 3D transforms to remove correlation in video sequence. The coding process is done in two stages: the first stage produces coarse sequence approximation (shaper) trying to fit as much information as possible in the limited bit budget. The second stage encodes the residual sequence, which is the difference between the original sequence and the shaper-reconstructed one. The shaper is obtained by pruned 3D-DCT, and the residual signal is coded by 3D-DCT or hybrid 3D transform. The redundancy is introduced by including the shaper in both descriptions. The amount of redundancy is easily controlled by the shaper quantization step. The scheme can also be easily optimized for suboptimal bit allocation. This optimization can run in real time during the encoding process.

The proposed MD video coder has low computational complexity, which makes it suitable for mobile devices with low computational power and limited battery life. The coder has been shown to outperform MDTC video coder and some simple MD coders based on H.263+. The coder performs especially well in a low-redundancy region. The encoder is also less computationally expensive than the H.263 encoder.

ACKNOWLEDGMENT

This work is supported by the Academy of Finland, Project no. 213462 (Finnish Centre of Excellence program (2006–2011)).

REFERENCES

[1] R. K. Chan and M. C. Lee, “3D-DCT quantization as a compression technique for video sequences,” in Proceedings of the Annual International Conference on Virtual Systems and Multimedia (VSMM ’97), pp. 188–196, Geneva, Switzerland, September 1997.
[2] S. Saponara, L. Fanucci, and P. Terreni, “Low-power VLSI architectures for 3D discrete cosine transform (DCT),” in Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS ’03), vol. 3, pp. 1567–1570, Cairo, Egypt, December 2003.
[3] A. Burg, R. Keller, J. Wassner, N. Felber, and W. Fichtner, “A 3D-DCT real-time video compression system for low complexity single-chip VLSI implementation,” in Proceedings of the Mobile Multimedia Conference (MoMuC ’00), p. 1B-5-1, Tokyo, Japan, November 2000.
[4] M. Bakr and A. E. Salama, “Implementation of 3D-DCT based video encoder/decoder system,” in Proceedings of the 45th IEEE Midwest Symposium on Circuits and Systems (MWSCAS ’02), vol. 2, pp. 13–16, Tulsa, Okla, USA, August 2002.
[5] S. Bousakta and H. O. Alshibami, “Fast algorithm for the 3-D DCT-II,” IEEE Transactions on Signal Processing, vol. 52, no. 4, pp. 992–1001, 2004.
[6] ITU-T Recommendation, Draft on H.263v2, 1999.
[7] J. J. Koivusaaari and J. H. Takala, “Simplified three-dimensional discrete cosine transform based video codec,” in Multimedia on Mobile Devices, vol. 5684 of Proceedings of SPIE, pp. 11–21, San Jose, Calif, USA, January 2005.
[8] V. K. Goyal, “Multiple description coding: compression meets the network,” IEEE Signal Processing Magazine, no. 5, pp. 74–93, 2001.
[9] J. G. Apostolopoulos and S. J. Wee, “Unbalanced multiple description video communication using path diversity,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’01), vol. 1, pp. 966–969, Thessaloniki, Greece, October 2001.
[10] A. R. Reibman, H. Jafarkhani, Y. Wang, M. T. Orchard, and R. Puri, “Multiple description coding for video using motion compensated prediction,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’99), vol. 3, pp. 837–841, Kobe, Japan, October 1999.
[11] J. G. Apostolopoulos, “Error-resilient video compression through the use of multiple states,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’00), vol. 3, pp. 352–355, Vancouver, BC, Canada, September 2000.
[12] V. Vaishampayan and S. A. John, “Balanced interframe multiple description video compression,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’99), vol. 3, pp. 812–816, Kobe, Japan, October 1999.
[13] Y. Wang, A. R. Reibman, and S. Lin, “Multiple description coding for video delivery,” Proceedings of the IEEE, vol. 93, no. 1, pp. 57–70, 2005.
[14] H. Man, R. L. de Queiroz, and M. J. T. Smith, “Three-dimensional subband coding techniques for wireless video communications,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 6, pp. 386–397, 2002.
[15] J. Kim, R. M. Mersereau, and Y. Altunbasak, “Error-resilient image and video transmission over the Internet using unequal error protection,” IEEE Transactions on Image Processing, vol. 12, no. 2, pp. 121–131, 2003.
[16] S. Somasundaram and K. P. Subbalakshmi, “3-D multiple description video coding for packet switched networks,” in Proceedings of IEEE International Conference on Multimedia and Expo (ICME ’03), vol. 1, pp. 589–592, Baltimore, Md, USA, July 2003.
[17] M. Yu, Z. Wenqin, G. Jiang, and Z. Yin, “An approach to 3D scalable multiple description video coding with content delivery networks,” in Proceedings of IEEE International Workshop on VLSI Design and Video Technology (IWVDVT ’05), pp. 191–194, Suzhou, China, May 2005.
[18] A. Norkin, A. Gotchev, K. Egiazarian, and J. Astola, “A low-complexity multiple description video coder based on 3D-transforms,” in Proceedings of the 14th European Signal Processing Conference (EUSIPCO ’06), Florence, Italy, September 2006.
[19] A. Norkin, A. Gotchev, K. Egiazarian, and J. Astola, “Two-stage multiple description image coders: analysis and comparative study,” Signal Processing: Image Communication, vol. 21, no. 8, pp. 609–625, 2006.
[20] A. M. Bruckstein, M. Elad, and R. Kimmel, “Down-scaling for better transform compression,” IEEE Transactions on Image Processing, vol. 12, no. 9, pp. 1132–1144, 2003.
[21] B.-L. Yeo and B. Liu, “Volume rendering of DCT-based compressed 3D scalar data,” IEEE Transactions on Visualization and Computer Graphics, vol. 1, no. 1, pp. 29–43, 1995.
[22] N. Bozinovic and J. Konrad, “Motion analysis in 3D DCT domain and its application to video coding,” Signal Processing: Image Communication, vol. 20, no. 6, pp. 510–528, 2005.

[23] H. S. Malvar and D. H. Staelin, “The LOT: transform coding without blocking effects,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 37, no. 4, pp. 553–559, 1989.

[24] W. H. R. Equitz and T. M. Cover, “Successive refinement of information,” IEEE Transactions on Information Theory, vol. 37, no. 2, pp. 269–275, 1991.

[25] A. N. Skodras, “Fast discrete cosine transform pruning,” IEEE Transactions on Signal Processing, vol. 42, no. 7, pp. 1833–1837, 1994.

[26] K. Rao and R. Yip, Discrete Cosine Transform: Algorithms, Advantages, Applications, Academic Press, London, UK, 1990.

[27] K. Yu, J. Lv, J. Li, and S. Li, “Practical real-time video codec for mobile devices,” in Proceedings of IEEE International Conference on Multimedia and Expo (ICME ‘03), vol. 3, pp. 509–512, Baltimore, Md, USA, July 2003.

[28] G. Cote, B. Erol, M. Gallant, and F. Kossentini, “H.263+: video coding at low bitrates,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 7, pp. 849–866, 1998.

[29] L. Roberts, “TMN 8 (h.263+) encoder/decoder, version 3.0,” 1997, Signal Processing and Multimedia Laboratory, University of British Columbia, Vancouver, BC, Canada, May, 1997.

[30] S. D. Kim, J. Yi, H. M. Kim, and J. B. Ra, “A deblocking filter with two separate modes in block-based video coding,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 1, pp. 156–160, 1999.

[31] D. Rusanovskyy and K. Egiazarian, “Post-processing for three-dimensional discrete cosine transform based video coding,” in Proceedings of the 7th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS ‘05), pp. 618–625, Antwerp, Belgium, September 2005.

[32] V. Vaishampayan, “Design of multiple description scalar quantizers,” IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 821–834, 1993.

[33] A. R. Reibman, H. Jafarkhani, Y. Wang, M. T. Orchard, and R. Puri, “Multiple-description video coding using motion-compensated temporal prediction,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 3, pp. 193–204, 2002.
Research Article

Energy-Efficient Acceleration of MPEG-4 Compression Tools

Andrew Kinane, Daniel Larkin, and Noel O’Connor

Centre for Digital Video Processing, Dublin City University, Glasnevin, Dublin 9, Ireland

Received 1 June 2006; Revised 21 December 2006; Accepted 6 January 2007

Recommended by Antonio Nunez

We propose novel hardware accelerator architectures for the most computationally demanding algorithms of the MPEG-4 video compression standard—motion estimation, binary motion estimation (for shape coding), and the forward/inverse discrete cosine transforms (incorporating shape adaptive modes). These accelerators have been designed using general low-energy design philosophies at the algorithmic/architectural abstraction levels. The themes of these philosophies are avoiding waste and trading area/performance for power and energy gains. Each core has been synthesised targeting TSMC 0.09 μm TCBN90LP technology, and the experimental results presented in this paper show that the proposed cores improve upon the prior art.

Copyright © 2007 Andrew Kinane et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Whilst traditional forms of frame-based video are challenging in their own right in this context, the situation becomes even worse when we look to future applications. In applications from multimedia messaging to gaming, users will require functionalities that simply cannot be supported with frame-based video formats, but that require access to the objects depicted in the content. Clearly this requires object-based video compression, such as that supported by MPEG-4, but this requires more complex and computationally demanding video processing. Thus, whilst object-video coding has yet to find wide-spread deployment in real applications, the authors believe that this is imminent and that this necessitates solutions for low-power object-based coding in the short term.

1.1. Object-based video

Despite the wider range of applications possible, object-based coding has its detractors due to the difficulty of the segmentation problem in general. However, it is the belief of the authors that in a constrained application such as mobile video telephony, valid assumptions simplify the segmentation problem. Hence certain object-based compression applications and associated benefits become possible. A screenshot of a face detection algorithm using simple RGB thresholding [1] is shown in Figure 1. Although video object segmentation is an open research problem, it is not the main focus of this work. Rather, this work is concerned with the problem of compressing the extracted video objects for efficient transmission or storage as discussed in the next section.

1.1.1. MPEG-4: object-based encoding

ISO/IEC MPEG-4 is the industrial standard for object-based video compression [2]. Earlier video compression standards encoded a frame as a single rectangular object, but MPEG-4 extends this to the semantic object-based paradigm. In MPEG-4 video, objects are referred to as video objects (VOs) and these are irregular shapes in general but may indeed represent the entire rectangular frame. A VO will evolve temporally at a certain frame rate and a snapshot of the state of a particular VO at a particular time instant is termed a video object plane (VOP). The segmentation (alpha) mask defines the shape of the VOP at that instant and this mask also evolves over time. A generic MPEG-4 video codec is similar in structure to the codec used by previous standards such as MPEG-1 and MPEG-2 but has additional functionality to support the coding of objects [3].

The benefits of an MPEG-4 codec come at the cost of algorithmic complexity. Profiling has shown that the most computationally demanding (and power consumptive) algorithms are, in order: ME, BME, and SA-DCT/IDCT [4–6].
A deterministic breakdown analysis is impossible in this instance because object-based MPEG-4 has content-dependent complexity. The breakdown is also highly dependent on the ME strategy employed. For instance, the complexity breakdown between ME, BME, and SA-DCT/IDCT is 66%, 13%, and 1.5% when encoding a specific test sequence using a specific set of codec parameters and full search ME with search window ±16 pixels [6]. The goal of the work presented in this paper is to implement these hotspot algorithms in an energy-efficient manner, which is vital for the successful deployment of an MPEG-4 codec on a mobile platform.

### 1.2. Low-energy design approach

Hardware architecture cores for computing video processing algorithms can be broadly classified into two categories: programmable and dedicated. It is generally accepted that dedicated architectures achieve the greatest silicon and power efficiency at the expense of flexibility [4]. Hence, the core architectures proposed in this paper (for ME, BME, SA-DCT, and SA-IDCT) are dedicated architectures. However, the authors argue that despite their dedicated nature, the proposed cores are flexible enough to be used for additional multimedia applications other than MPEG-4. This point is discussed in more detail in Section 6.

The low-energy design techniques employed for the proposed cores (see Sections 2–5) are based upon three general design philosophies.

1. Most savings are achievable at the higher levels of design abstraction since wider degrees of freedom exist [7, 8].
2. Avoid unnecessary computation and circuit switching [7].
3. Trade performance (in terms of area and/or speed) for energy gains [7].

Benchmarking architectures is a challenging task, especially if competing designs in the literature have been implemented using different technologies. Hence, to evaluate the designs proposed in this paper, we have used some normalisations to compare in terms of power and energy and a technology-independent metric to evaluate area and delay. Each of these metrics are briefly introduced here and are used in Sections 2–5.

#### 1.2.1. Product of gate count and computation cycles

The product of gate count and computation cycles (PGCC) for a design combines its latency and area properties into a single metric, where a lower PGCC represents a better implementation. The clock cycle count of a specific architecture for a given task is a fair representation of the delay when benchmarking, since absolute delay (determined by the clock frequency) is technology dependent. By the same rationale, gate count is a fairer metric for circuit area when benchmarking compared to absolute area in square millimetres.

#### 1.2.2. Normalised power and energy

Any attempt to normalise architectures implemented with two different technologies is effectively the same process as device scaling because all parameters must be normalised according to the scaling rules. The scaling formula when normalising from a given process $L$ to a reference process $L'$ is given by $L' = S 	imes L$, where $L$ is the transistor channel length. Similarly, the voltage $V$ is scaled by a factor $U$ according to $V' = U 	imes V$.

With the scaling factors established, the task now is to investigate how the various factors influence the power $P$. Using a first order approximation, the power consumption of a circuit is expressed as $P \propto CV^2f\alpha$, where $P$ depends on the capacitive load switched $C$, the voltage $V$, the operating frequency $f$, and the node switching probability $\alpha$. Further discussion about how each parameter scales with $U$ and $S$ can be found in [9]. This reference shows that normalising $P$ with respect to $\alpha$, $V$, $L$, and $f$ is achieved by (1),

$$P' = P \times S^2 \times U.$$  \hspace{1cm} (1)

With an expression for the normalised power consumption established by (1), the normalised energy $E'$ consumed by the proposed design with respect to the reference technology is expressed by (2), where $D$ is the absolute delay of the circuit to compute a given task and $C$ is the number of clock cycles required to compute that task,

$$E' = P' \times D = P' \times \frac{1}{f} \times C.$$  \hspace{1cm} (2)

Another useful metric is the energy-delay product (EDP), which combines energy and delay into a single metric. The normalised EDP is given by (3),

$$\text{EDP}' = P' \times D^2.$$  \hspace{1cm} (3)

This section has presented four metrics that attempt to normalise the power and energy properties of circuits for benchmarking. These metrics are used to benchmark the MPEG-4 hardware accelerators presented in this paper against prior art.
2. MOTION ESTIMATION

2.1. Algorithm

Motion estimation is the most computationally intensive MPEG-4 tool, requiring over 50% of the computational resources. Although different approaches to motion estimation are possible, in general the block-matching algorithm (BMA) is favoured. The BMA consists of two tasks: a block-matching task carrying out a distance criteria evaluation and a search task specifying the sequence of candidate blocks where the distance criteria is calculated. Numerous distance criteria for BMA have been proposed, with the sum-of-absolute-differences (SAD) criteria proved to deliver the best accuracy/complexity ratio particularly from a hardware implementation perspective [6].

2.2. Prior art review

Systolic-array (SA-) based architectures are a common solution proposed for block-matching-based ME. The approach offers an attractive solution, having the benefit of using memory bandwidth efficiently and the regularity allows significant control circuitry overhead to be eliminated [10]. Depending on the systolic structure, a SA implementation can be classified as one-dimensional (1D) or two-dimensional (2D), with global or local accumulation [11]. Clock rate, frame size, search range, and block size are the parameters used to decide on the number of PEs in the systolic structure [10].

The short battery life issue has most recently focused research on operation redundancy-free BM-based ME approaches. They are the so-called fast exhaustive search strategies and they employ conservative SAD estimations (thresholds) and SAD cancellation mechanisms [12, 13]. Furthermore, for heuristic (non-regular) search strategies (e.g., logarithmic searches), the complexity of the controller needed to generate data addresses and flow control signals increases considerably along with the power inefficiency. In order to avoid this, a tree-architecture BM is proposed in [14]. Nakayama et al. outline a hardware architecture for a heuristic scene adaptive search [15]. In many cases, the need for high video quality has steered low-power ME research to date, ensuring that the low throughput issues of a fully serial solution by employing pixel subsampling/remapping. Early termination of the SAD calculation is based on the premise that if the current block match has an intermediate SAD value exceeding that of the minimum SAD found so far, early termination is possible. In hardware implementations usage of this technique is rare [16], since the serial type processing required for SAD cancellation is not suited to SA architectures. Our proposed design uses SAD cancellation while avoiding the low throughput issues of a fully serial solution by employing pixel subsampling/remapping. In comparison to [16], which also implements early termination in a 2D SA architecture, the granularity of the SAD cancellation is far greater in our design. This will ultimately lead to greater dynamic power savings. While our approach employs 4 or 16 PEs, the 2D SA architecture uses 256 PEs in [16], hence roughly 64 to 16 times area savings are achieved with our architectures, respectively. As in any trade-off, these significant power and area savings are possible in our architectures at the expense of lower throughput (see Section 2.4). However, apart from the power-aware trade-off we propose with our architecture, another advantage is the fact that they can be reconfigured at run time to deal with variable block size, which is not the case for the SA architectures.

In order to carry out the early exit in parallel hardware, the SAD cancellation mechanism has to encompass both the

2.3. Proposed ME architecture

Early termination of the SAD calculation is based on the premise that if the current block match has an intermediate SAD value exceeding that of the minimum SAD found so far, early termination is possible. In hardware implementations usage of this technique is rare [16], since the serial type processing required for SAD cancellation is not suited to SA architectures. Our proposed design uses SAD cancellation while avoiding the low throughput issues of a fully serial solution by employing pixel subsampling/remapping. In comparison to [16], which also implements early termination in a 2D SA architecture, the granularity of the SAD cancellation is far greater in our design. This will ultimately lead to greater dynamic power savings. While our approach employs 4 or 16 PEs, the 2D SA architecture uses 256 PEs in [16], hence roughly 64 to 16 times area savings are achieved with our architectures, respectively. As in any trade-off, these significant power and area savings are possible in our architectures at the expense of lower throughput (see Section 2.4). However, apart from the power-aware trade-off we propose with our architecture, another advantage is the fact that they can be reconfigured at run time to deal with variable block size, which is not the case for the SA architectures.

In order to carry out the early exit in parallel hardware, the SAD cancellation mechanism has to encompass both the
is to employ block-level parallelism in the SAD formula (see (4)) and then transform the equation from calculating an absolute value \((6)\) to calculating a relative value to the current \(\text{min SAD} (7)\),

\[
\text{SAD} (MB_c, MB_r) = \sum_{j=1}^{16} \sum_{i=1}^{16} |MB_c(i,j) - MB_r(i,j)| = \sum_{k=0}^{3} \sum_{i=1}^{8} \sum_{j=1}^{8} |B_{c}(i,j) - B_{r}(i,j)| = \sum_{k=0}^{3} \text{BSAD}_k,
\]

\[
\text{min SAD} = \sum_{k=0}^{3} \text{min BSAD}_k, \quad (5)
\]

\[
\text{curr SAD} (MB_c, MB_r) = \sum_{k=0}^{3} \text{curr BSAD}_k, \quad (6)
\]

\[
\text{rel SAD} (MB_c, MB_r) = \text{min SAD} - \text{curr SAD} (MB_c, MB_r) = \sum_{k=0}^{3} (\text{min BSAD}_k - \text{curr BSAD}_k). \quad (7)
\]

Equation (5) gives the min SAD’s formula, calculated for the best match with (4). One should notice that the min BSAD\(_k\) values are not the minimum SAD values for the respective blocks. However, together they give the minimum SAD at MB-level. Min SAD and min BSAD\(_k\) are constant throughout the subsequent block matches (in (7)) until they are replaced by next best matches’ SAD values. Analysing (7) the following observations can be made. First, from a hardware point of view, the SAD cancellation comparison is implemented by de-accumulating instead of accumulating the absolute differences. Thus two operations (accumulation and comparison) can be implemented with only one operation (de-accumulation). Hence, anytime all block-level rel BSAD\(_k\) values are negative, it is obvious that a SAD cancellation condition has been met and one should proceed to the next match. Statistically, the occurrence of the early SAD cancellation is frequent (test sequence dependent) and therefore the calculation of the overall rel SAD value is seldom needed. Thus, in the proposed architecture the rel SAD update is carried out only if no cancellation occurred. Thus, if by the end of a match the SAD cancellation has not been met, only then rel SAD has to be calculated to see if globally (at MB level) the rel BSAD\(_k\) values give a better match (i.e., a negative rel SAD is obtained). During the update stage, if the rel SAD is negative, then no other update/correction is needed. However, if it is a better match, then the min SAD and min BSAD\(_k\) values have also to be updated. The new best match min BSAD\(_k\) values have also to be updated at block-level for the current and next matches. This is the function of the update stage. Second, it is clear intuitively from (7) that the smaller the min BSAD\(_k\) values are, the greater the probability for early SAD cancellation is. Thus, the quicker the SAD algorithm converges toward the best matches (i.e., smaller min BSAD\(_k\)), the more effective the SAD cancellation mechanism is at saving redundant operations. If SAD cancellation does not occur, all operations must be carried out. This implies that investigations should focus on motion prediction techniques and snail-type search strategies (e.g., circular, diamond) which start searching from the position that is most likely to be the best match, obtaining the smallest min BSAD\(_k\) values from earlier steps. Third, there is a higher probability (proved experimentally by this work) that the block level rel BSAD\(_k\) values become negative at the same time before the end of the match, if the blocks (B) are similar lower-resolution versions of the macroblock (MB). This can be achieved by remapping the video content as in Figure 2.
where the video frame is subsampled and partitioned in 4 subframes with similar content. Thus the ME memory (both for current block and search area) is organised in four banks that are accessed in parallel.

Figure 4 depicts a detailed view of a block-matching (BM) processing element (PE) proposed here. A SAD calculation implies a subtraction, an absolute, and an accumulation operation. Since only values relatives to the current min SAD and min BSAD values are calculated, a de-accumulation function is used instead. The absolute difference is de-accumulated from the DACC_REG register (deaccumulator).

At each moment, the DACC_REG stores the appropriate rel BSAD value and signals immediately with its sign bit if it becomes negative. The initial value stored in the DACC_REG at the beginning of each match is the corresponding min BSAD value and is brought through the local_SAD_val inputs. Whenever all the DACC_REG de-accumulate become negative they signal a SAD cancellation condition and the update stage is kept idle.

The update stage is carried out in parallel with the next match’s operations executed in the block-level datapaths because it takes at most 11 cycles. Therefore, a pure sequential scheduling of the update stage operations is implemented in the update stage hardware (Figure 3). There are three possible update stage execution scenarios: first, when it is idle most of the time, second, when the update is launched at the end of a match, but after 5 steps the global rel SAD turns out to be negative and no update is deemed necessary (see Figure 5(a)), third, when after 5 steps rel SAD is positive (see Figure 5(b)). In the latter case, the min SAD and min BSAD values, stored, respectively, in TOT_MIN_SAD_REG and BSAD_REG, are updated. Also, the rel BSAD corrections, stored beforehand in the PREV_DACC_REG registers, have to be made to the PEs’ DACC_REG registers. The correction operation involves a subtraction of the PREV_DACC_REG values (inverters provided in Figure 3 to obtain 2’s complement) from the DACC_REG registers through the prev_dacc_val inputs of the BM PEs. There is an extra cycle added for the correction operation, when the PE halts the normal de-accumulation function. These corrections change the min SAD and min BSAD values, thus the PEs should have started with in the new match less than 11 cycles ago. One should also note that if a new SAD cancellation occurs and a new match is skipped, this does not affect the update stage’s operations. That is due to the fact that a match skip means that the resulting curr SAD value was getting larger than the current min SAD which can only be updated with a smaller value. Thus, the match skip would have happened even if the min SAD value had been updated already before the start of the current skipped match.

2.4. Experimental results

A comparison in terms of operations and cycles between our adaptive architecture (with a circular architecture, a 16 × 16 MB and a search window of ±7 pels) and two SA architectures (a typical 1D SA architecture and a 2D SA architecture [16]) is carried out in this section. Results are presented for a variety of MPEG QCIF test sequences. Table 1 shows that our early termination architecture outperforms a typical 1D SA architecture. The 4 × PE succeeds in cancelling the largest number of SAD operations (70% average reduction for the sequences listed in Table 1), but at the price of a longer execution time (i.e., larger number of cycles) for videos that exhibit high levels of motion (e.g., the MPEG Foreman test sequence). The 16 × PE outperforms the 1D SA both for the number of SAD operations and for the total number of cycles (i.e., execution time). In comparison with the 4 × PE architecture, the 16 × PE architecture is faster but removes less redundant SAD operations. Thus, choosing between 4 × PE and 16 × PE is a trade-off between processing speed and power savings. With either architecture, to cover scenarios where there is below average early termination (e.g., Foreman sequence), the operating clock frequency is set to a frequency which includes a margin that provides adequate throughput for natural video sequences.

In comparison with the 2D SA architecture proposed in [16], our architecture outperforms in terms of area and switching (SAD operations) activity. A pipelined 2D SA architecture as the one presented in [16] executes the 1551 million SAD operations in approximately 13 million clock cycles. The architecture in [16] pays the price of disabling the switching for up to 45% of the SAD operations by employing extra logic (requiring at least 66 adders/subtracters), to
carry out a conservative SAD estimation. With 4 PEs and 16 PEs, respectively, our architectures are approximately 64 and 16 times smaller (excluding the conservative SAD estimation logic). In terms of switching, special latching logic is employed to block up to 45% of the SAD operation switching. This is on average less than the number of SAD operations cancelled by our architectures. In terms of throughput, our architectures are up to 10 times slower than the 2D SA architecture proposed in [16], but for slow motion test sequences (e.g., akiyo), the performance is very much comparable. Hence, we claim that the trade-off offered by our architectures is more suitable to power-sensitive mobile devices.

The ME 4 × PE design was captured using Verilog HDL and synthesised using Synopsys Design Compiler, targeting a TSMC 90 nm library characterised for low power. The resultant area was 7.5 K gates, with a maximum possible operating frequency $f_{\text{max}}$ of 700 MHz. The average power consumption for a range of video test sequences is 1.2 mW (@100 MHz, 1.2 V, 25°C). Using the normalisations presented in Section 1.2.2, it is clear from Table 2 that the normalised power ($P'$) and energy ($E'$) of Takahashi et al. [17] and Nakayama et al. [15] are comparable to the proposed architecture. The fact that the normalised energies of all three approaches are comparable is interesting, since both Takahashi and Nakayama use fast heuristic search strategies, whereas the proposed architecture uses a fast-exhaustive approach based on SAD cancellation. Nakayama have a better normalised EDP but they use only the top four bits of each pixel when computing the SAD, at the cost of image quality. The fast-exhaustive approach has benefits such as more regular memory access patterns and smaller prediction residuals (better PSNR). The latter benefit has power consequences for the subsequent transform coding, quantisation and entropy coding of the prediction residual.

### 3. BINARY MOTION ESTIMATION

#### 3.1. Algorithm

Similar to texture pixel encoding, if a binary alpha block (BAB) belongs to a MPEG-4 inter video object plane (P-VOP), temporal redundancy can be exploited through the use of motion estimation. However, it is generally accepted that motion estimation for shape is the most computationally intensive block within binary shape encoding [18]. Because of this computational complexity hot spot, we leverage and extend our work on the ME core to carry out BME processing in a power-efficient manner.

The motion estimation for shape process begins with the generation of a motion vector predictor for shape (MVPS) [19]. The predicted motion compensated BAB is retrieved and compared against the current BAB. If the error between each $4 \times 4$ sub block of the predicted BAB and the current BAB is less than a predefined threshold, the motion vector predictor can be used directly [19]. Otherwise an accurate motion vector for shape (MVS) is required. MVS is a conventional BME process. Any search strategy can be used and typically a search window size of ±16 pixels around the MVPS BAB is employed.

#### 3.2. Prior art review

Yu et al. outline a software implementation for motion estimation for shape, which uses a number of intermediate thresholds in a heuristic search strategy to reduce the computational complexity [20]. We do not consider this approach viable for a hardware implementation due to the irregular memory addressing, in addition to providing limited scope for exploiting parallelism.
Boundary mask methods can be employed in a preprocessing manner to reduce the number of search positions [21, 22]. The mask generation method proposed by Panu sopone and Chen, however, is computational intensive due to the block loop process [21]. Tsai and Chen use a more efficient approach [22] and present a proposed hardware architecture. In addition Tsai et al. use heuristics to further reduce the search positions. Chang et al. use a 1D systolic array architecture coupled with a full search strategy for the BME implementation [18]. Improving memory access performance is a common optimisation in MPEG-4 binary shape encoders [23, 24]. Lee et al. suggest a run length coding scheme to minimise on-chip data transfer and reduce memory requirements, however the run length codes still need to be decoded prior to BME [24].

Our proposed solution leverages our ME SAD cancellation architecture and extends this by avoiding unnecessary operations by exploiting redundancies in the binary shape information. This is in contrast to a SA approach, where unnecessary calculations are unavoidable due to the data flow in the systolic structure. Unlike the approach of Tsai and Chen, we use an exhaustive search to guarantee finding the best block match within the search range [22].

### 3.3. Proposed BME architecture

When using binary-valued data the ME SAD operation simplifies to the form given in (8), where $B_{cur}$ is the BAB under consideration in the current binary alpha plane (BAP) and $B_{ref}$ is the BAB at the current search location in the reference BAP,

$$\text{SAD}(B_{cur}, B_{ref}) = \sum_{i=1}^{16} \sum_{j=1}^{16} B_{cur}(i,j) \odot B_{ref}(i,j).$$

In previous BME research, no attempts have been made to optimise the SAD PE datapath. However, the unique characteristics of binary data mean further redundancies can be exploited to reduce datapath switching activity. It can be seen from (8) that there are unnecessary memory accesses and operations when both $B_{cur}$ and $B_{ref}$ pixels have the same value, since the XOR will give a zero result. To minimise this effect, we propose reformulating the conventional SAD equation.

The following properties can be observed from Figure 6(a):

$$\text{TOTAL}_{\text{cur}} = \text{COMMON} + \text{UNIQUE}_{\text{cur}},$$

$$\text{TOTAL}_{\text{ref}} = \text{COMMON} + \text{UNIQUE}_{\text{ref}},$$

where

- (a) $\text{TOTAL}_{\text{cur}}$ is the total number of white pixels in the current BAB.
- (b) $\text{TOTAL}_{\text{ref}}$ is the total number of white pixels in the reference BAB.
- (c) COMMON is the number of white pixels that are common in both the reference BAB and the current BAB.
- (d) UNIQUE$_{\text{cur}}$ is the number of white pixels in the current BAB but not in the reference BAB.
- (e) UNIQUE$_{\text{ref}}$ is the number of white pixels in the reference block but not in the current BAB.

It is also clear from Figure 6(a), that the SAD value between the current and reference BAB can be represented as

$$\text{SAD} = \text{UNIQUE}_{\text{cur}} + \text{UNIQUE}_{\text{ref}}.$$  \hspace{1cm} (10)

Using these identities, it follows that

$$\text{SAD} = \text{TOTAL}_{\text{ref}} - \text{TOTAL}_{\text{cur}} + 2 \times \text{UNIQUE}_{\text{cur}}.$$  \hspace{1cm} (11)

Equation (11) can be intuitively understood as $\text{TOTAL}_{\text{ref}} - \text{TOTAL}_{\text{cur}}$ being a conservative estimate of the SAD value, whilst $2 \times \text{UNIQUE}_{\text{cur}}$ is an adjustment to the conservative SAD estimate to give the correct final SAD value. The reason equation (11) is beneficial is because the following.

- (a) $\text{TOTAL}_{\text{cur}}$ is calculated only once per search.
- (b) $\text{TOTAL}_{\text{ref}}$ can be updated in 1 clock cycle, after initial calculation, provided a circular search is used.
- (c) Incremental addition of $\text{UNIQUE}_{\text{cur}}$ allows early termination if the current minimum SAD is exceeded.
- (d) Whilst it is not possible to know $\text{UNIQUE}_{\text{cur}}$ in advance of a block match, run length coding can be used to encode the position of the white pixels in the current BAB, thus minimising access to irrelevant data.

Run length codes (RLC) are generated in parallel with the first block match of the search window, an example of typical RLC is illustrated in Figure 7. It is possible to do the run length encoding during the first match, because early termination of the SAD calculation is not possible at this stage, since a minimum SAD has not been found. The first match

---

**Table 2: ME synthesis results and benchmarking.**

| Architecture       | Tech (µm) | Cycle count | Gates | PGCC | f (MHz) | Power (mW) | $P'$ (mW) | $E'$ (nJ) | EDP' (fls) |
|--------------------|----------|-------------|-------|------|---------|------------|-----------|----------|------------|
| Takahashi et al. [17] | 0.25     | 32 768 n/a  | 16 384 n/a | n/a  | 60      | 2.8        | 0.3       | 81       | 22 401     |
| Nakayama et al. [15]  | 0.18     | n/a n/a 300 | n/a n/a  | n/a  | 250     | 9.0        | 1.8       | 40       | 889        |
| Proposed            | 0.09     | 16 384 574 | 3618 7577 | 2.74 × 10^7 | 100     | 1.2        | 1.2       | 43       | 1508       |

---

Andrew Kinane et al. 7

\[21, 22\]. The mask generation method proposed by Panu sopone and Chen, however, is computational intensive due to the block loop process [21]. Tsai and Chen use a more efficient approach [22] and present a proposed hardware architecture. In addition Tsai et al. use heuristics to further reduce the search positions. Chang et al. use a 1D systolic array architecture coupled with a full search strategy for the BME implementation [18]. Improving memory access performance is a common optimisation in MPEG-4 binary shape encoders [23, 24]. Lee et al. suggest a run length coding scheme to minimise on-chip data transfer and reduce memory requirements, however the run length codes still need to be decoded prior to BME [24].

Our proposed solution leverages our ME SAD cancellation architecture and extends this by avoiding unnecessary operations by exploiting redundancies in the binary shape information. This is in contrast to a SA approach, where unnecessary calculations are unavoidable due to the data flow in the systolic structure. Unlike the approach of Tsai and Chen, we use an exhaustive search to guarantee finding the best block match within the search range [22].

### 3.3. Proposed BME architecture

When using binary-valued data the ME SAD operation simplifies to the form given in (8), where $B_{cur}$ is the BAB under consideration in the current binary alpha plane (BAP) and $B_{ref}$ is the BAB at the current search location in the reference BAP,

$$\text{SAD}(B_{cur}, B_{ref}) = \sum_{i=1}^{16} \sum_{j=1}^{16} B_{cur}(i,j) \odot B_{ref}(i,j).$$

In previous BME research, no attempts have been made to optimise the SAD PE datapath. However, the unique characteristics of binary data mean further redundancies can be exploited to reduce datapath switching activity. It can be seen from (8) that there are unnecessary memory accesses and operations when both $B_{cur}$ and $B_{ref}$ pixels have the same value, since the XOR will give a zero result. To minimise this effect, we propose reformulating the conventional SAD equation.

The following properties can be observed from Figure 6(a):

$$\text{TOTAL}_{\text{cur}} = \text{COMMON} + \text{UNIQUE}_{\text{cur}},$$

$$\text{TOTAL}_{\text{ref}} = \text{COMMON} + \text{UNIQUE}_{\text{ref}},$$

where

- (a) $\text{TOTAL}_{\text{cur}}$ is the total number of white pixels in the current BAB.
- (b) $\text{TOTAL}_{\text{ref}}$ is the total number of white pixels in the reference BAB.
- (c) COMMON is the number of white pixels that are common in both the reference BAB and the current BAB.
- (d) UNIQUE$_{\text{cur}}$ is the number of white pixels in the current BAB but not in the reference BAB.
- (e) UNIQUE$_{\text{ref}}$ is the number of white pixels in the reference block but not in the current BAB.

It is also clear from Figure 6(a), that the SAD value between the current and reference BAB can be represented as

$$\text{SAD} = \text{UNIQUE}_{\text{cur}} + \text{UNIQUE}_{\text{ref}}.$$  \hspace{1cm} (10)

Using these identities, it follows that

$$\text{SAD} = \text{TOTAL}_{\text{ref}} - \text{TOTAL}_{\text{cur}} + 2 \times \text{UNIQUE}_{\text{cur}}.$$  \hspace{1cm} (11)

Equation (11) can be intuitively understood as $\text{TOTAL}_{\text{ref}} - \text{TOTAL}_{\text{cur}}$ being a conservative estimate of the SAD value, whilst $2 \times \text{UNIQUE}_{\text{cur}}$ is an adjustment to the conservative SAD estimate to give the correct final SAD value. The reason equation (11) is beneficial is because the following.

- (a) $\text{TOTAL}_{\text{cur}}$ is calculated only once per search.
- (b) $\text{TOTAL}_{\text{ref}}$ can be updated in 1 clock cycle, after initial calculation, provided a circular search is used.
- (c) Incremental addition of $\text{UNIQUE}_{\text{cur}}$ allows early termination if the current minimum SAD is exceeded.
- (d) Whilst it is not possible to know $\text{UNIQUE}_{\text{cur}}$ in advance of a block match, run length coding can be used to encode the position of the white pixels in the current BAB, thus minimising access to irrelevant data.

Run length codes (RLC) are generated in parallel with the first block match of the search window, an example of typical RLC is illustrated in Figure 7. It is possible to do the run length encoding during the first match, because early termination of the SAD calculation is not possible at this stage, since a minimum SAD has not been found. The first match
always takes $N \times N$ (where $N$ is the block size) cycles to complete and this provides ample time for the run length encoding process to operate in parallel. After the RLC encoding, the logic can be powered down until the next current block is processed.

In situations where there are fewer black pixels than white pixels in the current MB or where $\text{TOTAL}_{\text{cur}}$ is greater than $\text{TOTAL}_{\text{ref}}$, (12) is used instead of (11). Since run length coding of the reference BAB is not feasible, $\text{UNIQUE}_{\text{ref}}$ can be generated by examining the black pixels in the current BAB. The location of the black pixels can be automatically derived from the RLC for the white pixels (see Figure 7). Thus, by reusing the RLC associated with the white pixels, additional memory is not required and furthermore the same SAD datapath can be reused with minimal additional logic,

$$\text{SAD} = \text{TOTAL}_{\text{cur}} - \text{TOTAL}_{\text{ref}} + 2 \times \text{UNIQUE}_{\text{ref}}.$$

Figure 6(b) shows a detailed view of the BME SAD PE. At the first clock cycle, the minimum SAD encountered so far is loaded into $\text{DACC}_\text{REG}$. During the next cycle $\text{TOTAL}_{\text{cur}}/\text{TOTAL}_{\text{ref}}$ is added to $\text{DACC}_\text{REG}$ (depending if $\text{TOTAL}_{\text{ref}}[\text{MSB}]$ is 0 or 1, respectively, or if $\text{TOTAL}_{\text{ref}}$ is larger than $\text{TOTAL}_{\text{cur}}$). On the next clock cycle, $\text{DACC}_\text{REG}$ is de-accumulated by $\text{TOTAL}_{\text{ref}}/\text{TOTAL}_{\text{cur}}$ again depending on whether $\text{TOTAL}_{\text{ref}}[\text{MSB}]$ is 0 or 1, respectively. If a sign change occurs at this point, the minimum SAD has already been exceeded and no further processing is required. If a sign change has not occurred, the address generation unit retrieves the next RLC from memory. This is decoded to give an $X, Y$ macroblock address. The $X, Y$ address is used to retrieve the relevant pixel from the reference MB and the current MB. The pixel values are XORed and the result is left shifted...
by one place and then subtracted from the DACC_REG. If a sign change occurs, early termination is possible. If not the remaining pixels in the current run length code are processed. If the SAD calculation is not cancelled, subsequent run length codes for the current MB are fetched from memory and the processing repeats.

When a SAD has been calculated or terminated early, the address generation unit moves the reference block to a new position. Provided a circular or full search is used, TOTALref can be updated in one clock cycle. This is done by subtracting the previous row or column (depending on search window movement) from TOTALref and adding the new row or column, this is done via a simple adder tree.

In order to exploit SAD cancellation, an intermediate partial SAD must be generated. This requires SAD calculation to proceed in a sequential manner, however this reduces encoding throughput and is not desirable for real time applications. To increase throughput parallelism must be exploited. Therefore, we leverage our ME approach and partition the BAB into four 8 × 8 blocks by using a simple pixel subsampling technique. Four PEs, each operating on one 8 × 8 block, generate four partial SAD values. The control logic uses these partially accumulated SAD values to make an overall SAD cancellation decision. If SAD cancellation does not occur and all alpha pixels in the block are processed, the update stage is evoked. The update logic is identical to the ME unit. Similar to the ME architecture, 16 PE can also be used, albeit at the expense of reduced cancellation.

### 3.4. Experimental results

Table 3 summarises the synthesis results for the proposed BME architecture using 4 PE. Synthesising the design with Synopsys Design Compiler targeting TSMC 0.09 μm TCBN90LP technology yields a gate count of 10 117 and a maximum theoretical operating frequency $f_{\text{max}}$ of 700 MHz. Unlike the constant throughput SA approaches, the processing latency to generate one set of motion vectors for the proposed architecture is data dependant. The worst and best case processing latencies are 65 535 and 3133 clock cycles, respectively. Similar to our ME architecture, the clock frequency includes a margin to cover below average early termination. As reported in our prior work [26], we achieve on average 90% early termination using common test sequences. Consequently this figure is used in the calculation of the PGCC ($6.63 \times 10^7$). BME benchmarking is difficult due to a lack of information in prior art, this includes BME architectures used in MPEG-4 binary shape coding and BME architectures used in low complexity approaches for texture ME [18, 22, 23, 25, 27].

The SA BME architecture proposed by Natarajan et al., is leveraged in the designs proposed by Chang et al. and Lee et al. Consequently similar cycle counts can be observed in each implementation [18, 23, 25]. The average cycle counts (6553 cycles) for our architecture is longer than the architecture proposed by Chang et al. [18], this is due to our architectural level design decision to trade off throughput for reduced SAD operations and consequently reduced power consumption. As a consequence of the longer latency, the PGCC for our proposed architecture is inferior to that of the architecture proposed by Chang et al. [18]. However, the PGCC metric does not take into account the nonuniform switching in our proposed design. For example, after the first block match the run length encoder associated with each PE is not active, in addition the linear pixel addressing for the first block match is replaced by the run length decoded pixel scheme for subsequent BM within the search window. The power, energy, and EDP all take account of the nonuniform data-dependant processing, however, benchmarking against prior art using these metrics is not possible due to a lack of information in the literature.

### 4. SHAPE ADAPTIVE DCT

#### 4.1. Algorithm

When encoding texture, an MPEG-4 codec divides each rectangular video frame into an array of nonoverlapping 8 × 8 texture blocks and processes these sequentially using the SA-DCT [28]. For blocks that are located entirely inside the VOP, the SA-DCT behaves identically to the 8 × 8 DCT. Any blocks located entirely outside the VOP are skipped to save needless processing. Blocks that lie on the VOP boundary (e.g., Figure 8) are encoded depending on their shape and only the opaque pixels within the boundary blocks are actually coded.

The additional factors that make the SA-DCT more computationally complex with respect to the 8 × 8 DCT are vector shape parsing, data alignment, and the need for a variable N-point 1D DCT transform. The SA-DCT is less regular compared to the 8 × 8 block-based DCT since its processing decisions are entirely dependent on the shape information associated with each individual block.
4.2. Prior art review

Le and Glesner have proposed two SA-DCT architectures—a reconfigurable structure and a feed-forward structure [29]. The authors favour the feed-forward architecture and this has a hardware cost of 11 adders and 5 multipliers, with a cycle latency of \(N + 2\) for an \(N\)-point DCT. However, neither of the architectures address the horizontal packing required to identify the lengths of the horizontal transforms and have the area and power disadvantage of using expensive hardware multipliers.

Tseng et al. propose a reconfigurable pipeline that is dynamically configured according to the shape information [30]. The architecture is hampered by the fact that the entire \(8 \times 8\) shape information must be parsed to configure the datapath “contexts” prior to texture processing.

Chen et al. developed a programmable datapath that avoids multipliers by using canonic signed digit (CSD) adder-based distributed arithmetic [31, 32]. The hardware cost of the datapath is 3100 gates requiring only a single adder, which is reused recursively when computing multiply-accumulates. This small area is traded-off against cycle latency—1904 in the worst case scenario. The authors do not comment on the perceptual performance degradation or otherwise caused by approximating odd length DCTs with even DCTs.

Lee et al. considered the packing functionality requirement and developed a resource shared datapath using adders and multipliers coupled with an autoaligning transpose memory [33]. The datapath is implemented using 4 multipliers and 11 adders. The worst case computation cycle latency is 11 clock cycles for an 8-point 1D DCT. This is the most advanced implementation, but the critical path caused by the multipliers in this architecture limits the maximum operating frequency and has negative power consumption consequences.

4.3. Proposed SA-DCT architecture

The SA-DCT architecture proposed in this paper tackles the deficiencies of the prior art by employing a reconfiguring adder-only-based distributed arithmetic structure. Multipliers are avoided for area and power reasons [32]. The top-level SA-DCT architecture is shown in Figure 9, comprising of the transpose memory (TRAM) and datapath with their associated control logic. For all modules, local clock gating is employed based on the computation being carried out to avoid wasted power.

It is estimated that an \(m\)-bit Booth multiplier costs approximately 18–20 times the area of an \(m\)-bit ripple carry adder [32]. In terms of power consumption, the ratio of multiplier power versus adder power is slightly smaller than area ratio since the transition probabilities for the individual nodes are different for both circuits. For these reasons, the architecture presented here is implemented with adders only.

4.3.1. Memory and control architecture

The primary feature of the memory and addressing modules in Figure 9 is that they avoid redundant register switching and latency when addressing data and storing intermediate values by manipulating the shape information. The addressing and control logic (ACL) parses shape and pixel data from an external memory and routes the data to the variable \(N\)-point 1D DCT datapath for processing in a column-wise fashion. The intermediate coefficients after the horizontal processing are stored in the TRAM. The ACL then reads each vertical data vector from this TRAM for horizontal transformation by the datapath.

The ACL has a set of pipelined data registers (BUFFER and CURRENT) that are used to buffer up data before routing to the variable \(N\)-point DCT datapath. There are also a set of interleaved modulo-8 counters (\(N_{\text{buff}_A, r}\) and \(N_{\text{buff}_B, r}\)). Each counter either stores the number of VOP pels in BUFFER or in CURRENT, depending on a selection signal. This pipelined/interleaved structure means that as soon as the data in CURRENT has completed processing, the next data vector has been loaded into BUFFER with its shape parsed. It is immediately ready for processing, thereby maximising throughput and minimising overall latency.

Data is read serially from the external data bus if in vertical mode or from the local TRAM if in horizontal mode. In vertical mode, when valid VOP pixel data is present on the input data bus, it is stored in location BUFFER[N_{\text{buff}_i, r}] in the next clock cycle (where \(i \in \{A, B\}\) depends on the interleaved selection signal). The 4-bit register \(N_{\text{buff}_i, r}\) is also incremented by 1 in the same cycle, which represents the number of VOP pels in BUFFER (i.e., the vertical \(N\) value). In this way vertical packing is done without redundant shift cycles and unnecessary power consumption. In horizontal mode, a simple FSM is used to address the TRAM. It using the \(N\) values already parsed in the vertical process.
to minimise the number of accesses. The same scheme ensures horizontal packing is done without redundant shift cycles.

The TRAM is a 64-word × 15-bit RAM that stores the coefficients produced by the datapath when computing the vertical 1D transforms. These coefficients are then read by the ACL in a transposed fashion and horizontally transformed by the datapath yielding the final SA-DCT coefficients. When storing data, the coefficient index \( k \) is manipulated to store the value at address \( 8 \times k + \text{NRAM}[k] \). Then \( \text{NRAM}[k] \) is incremented by 1. In this way, when an entire block has been vertically transformed, the TRAM has the resultant data stored in a horizontally packed manner with the horizontal \( N \) values ready in NRAM immediately without shifting. These \( N \) values are used by the ACL to minimise TRAM reads for the horizontal transforms.

### 4.3.2. Datapath architecture

The variable \( N \)-point 1D DCT module is shown in Figure 10, which computes all \( N \) coefficients serially starting with \( F[N-1] \) down to \( F[0] \). This is achieved using even/odd decomposition (EOD), followed by adder-based distributed arithmetic using a multiplexed weight generation module (MWGM) and a partial product summation tree (PPST). A serial coefficient computation scheme was chosen because of the adaptive nature of the computations and the shape parsing logic is simpler this way.

EOD exploits the inherent symmetries in the SA-DCT cosine basis functions to reduce the complexity of the subsequent MWGM computation. The EOD module (Figure 10) decomposes the input vector and reuses the same adders for both even and odd \( k \). This adder reuse requires MUXs but...
the savings in terms of adders offsets this and results in an overall area improvement with only a slight increase in critical path delay. Register clocking of s and d is controlled so that switching only occurs when necessary.

The dot product of the (decomposed) data vector with the appropriate N-point DCT basis vector yielding SA-DCT coefficient k is computed using a reconfiguring adder-based distributed arithmetic structure (the MWGM) followed by a PPST as shown in Figure 10. Using dedicated units for different \( \{k,N\} \) combinations (where at most only one will be active at any instant) is avoided by employing a reconﬁguring multiplexing structure based on \( \{k,N\} \) that reuses single resources. Experimental results have shown that for a range of video test sequences, 13 distributed binary weights are needed to adequately satisfy reconstructed image quality requirements [34]. The adder requirement (11 in total) for the 13-weight MWGM has been derived using a recursive iterative matching algorithm [35].

The datapath of the MWGM is conﬁgured to compute the distributed weights for N-point DCT coeﬃcient k using the 6-bit vector \( \{k,N\} \) as shown in Figure 10. Even though \( 0 \leq N \leq 8 \), the case of \( N = 0 \) is redundant so the range \( 1 \leq N \leq 8 \) can be represented using three bits (range \( 0 \leq k \leq N - 1 \) also requires three bits). Even though the select signal is 6 bits wide, only 36 cases are valid since the 28 cases where \( k \geq N \) do not make sense so the MUX logic complexity is reduced. For each of the weights, there is a certain degree of equivalence between subsets of the 36 valid cases which again decreases the MUX complexity. Signal \( \overline{even/odd} \) (equivalent to the LSB of \( k \)) selects the even or odd decomposed data vector and the selected vector (signals \( x0, x1, x2, \) and \( x3 \) in Figure 10) drive the 11 adders. Based on \( \{k,N\} \), the MUXs select the appropriate value for each of the 13 weights. There are 16 possible values (zero and signals \( x0, x1, \ldots, x12x3 \) in Figure 10) although each weight only chooses from a subset of these possibilities. The weights are then combined by the PPST to produce coeﬃcient \( F(k) \).

Again, power consumption issues have been considered by providing a valid signal that permits the data in the weight registers to only switch when the control logic flags it necessary. The logic paths have been balanced in the implementation in the sense that the delay paths from each of the MWGM input ports to the data input of the weight registers are as similar as possible. This has been achieved by designing the adders and multiplexers in a tree structure as shown in Figure 10, reducing the probability of net glitching when new data is presented at the input ports.

The use of adder-based distributed arithmetic necessitates a PPST to combine the weights together to form the final coeﬃcient as shown in Figure 10. Since it is a potential critical path, a carry-save Wallace tree structure using \((3 : 2)\) counters and \((4 : 2)\) compressors has been used to postpone carry propagation until the ﬁnal ripple carry addition. The weighted nature of the inputs means that the sign extension can be manipulated to reduce the circuit complexity of the high order compressors [36]. Vertical coeﬃcients are rounded to 11, \( f \) ﬁxed-point bits (11 bits for integer and \( f \) bits for fractional) and our experiments show that \( f = 4 \) represents a good trade-oﬀ between area and performance. This implies that each word in the TRAM is 15 bits wide. Horizontal coeﬃcients are rounded to 12.0 bits and are routed directly to the module outputs.

### 4.4. Experimental results

Table 4 summarises the synthesis results for the proposed SA-DCT architecture and the normalised power and energy metrics to facilitate a comparison with prior art. Synthesising the design with Synopsys Design Compiler targeting TSMC 0.09 \( \mu m \) TCBN90LP technology yields a gate count of 25 583 and a maximum theoretical operating frequency \( f_{\text{max}} \) of 556 MHz. The area of the variable N-point 1D DCT datapath is 12 016 (excluding TRAM memory and ACL). Both gate counts are used to facilitate equivalent benchmarking with other approaches based on the information available in the literature. The results show the proposed design is an improvement over Lee [33] and offers a better trade-oﬀ in terms of cycle count versus area compared with Chen [32], as discussed subsequently. Area and power consuming multipliers have been eliminated by using only adders—27 in total—divided between the EOD module (4), the MWGM (11) and the PPST (12). Using the estimation that a mutliplier is equivalent to about 20 adders in terms of area,
the adder count of the proposed architecture (27) compares favourably with Le [29] \((5 \times 20 + 11 = 111)\) and Lee [33] \((4 \times 20 + 11 = 91)\). This is offset by the additional MUX overhead, but as evidenced by the overall gate count figure of the proposed architecture, it still yields an improved circuit area. By including the TRAM (1) and the ACL controller (3), an additional 4 adders are required by the entire proposed design. In total, the entire design therefore uses 31 adders and no multipliers. In terms of area and latency, the PGCC metric shows that the proposed architecture outperforms the Chen [32] and Lee [33] architectures.

The power consumption figure of 0.36 mW was obtained by running back-annotated dynamic simulation of the gate level netlist for various V0 sequences and taking an average (@11 MHz, 1.2 V, 25°C). The simulations were run at 11 MHz since this is the lowest possible operating frequency that guarantees 30 fps CIF real-time performance, given a worst case cycle latency per block of 142 cycles. The Synopsys Prime Power tool is used to analyse the annotated switching information from a VCD file. Only two of the SA-DCT implementations in the literature quote power consumption figures and the parameters necessary against which to perform normalised benchmarking: the architectures by Tseng et al. [30] and Chen et al. [32]. The normalised power, energy, and energy-delay product (EDP) figures are summarised in Table 4. Note that the energy figures quoted in the table are the normalised energies required to process a single 8 \(\times\) 8 block. Results show that the proposed SA-DCT architecture compares favourably against both the Tseng and the Chen architectures. The normalised energy dissipation and EDP figures are the most crucial in terms of benchmarking, since the energy dissipated corresponds to the amount of drain on the battery and the lifetime of the device.

5. SHAPE-ADAPTIVE IDCT

5.1. Algorithm

The SA-IDCT reverses the SA-DCT process in the feedback loop of a video encoder and also in the decoder. The starting point for the SA-IDCT is a block of coefficients (that have been computed by the SA-DCT) and a shape/alpha block corresponding to the pattern into which the reconstructed pixels will be arranged. The SA-IDCT process begins by parsing the 8 \(\times\) 8 shape block so that the coefficients can be addressed correctly and the pixels can be reconstructed in the correct pattern. Based on the row length (0 \(\leq N \leq 8\)), a 1D N-point IDCT for each row of coefficients is calculated. Subsequently, the produced intermediate horizontal results are realigned to their correct column according to the shape block and a 1D N-point IDCT for each column is performed. Finally, the reconstructed pixels are realigned vertically to their original VOP position.

The additional factors that make the SA-IDCT more computationally complex with respect to the 8 \(\times\) 8 IDCT are vector shape parsing, data alignment, and the need for a variable N-point 1D IDCT transform. The SA-IDCT is less regular compared to the 8 \(\times\) 8 block-based IDCT since its processing decisions are entirely dependent on the shape information associated with each individual block. Also, peculiarities of the SA-IDCT algorithm mean that the shape parsing and data alignment steps are more complicated compared to the SA-DCT.

5.2. Prior art review

The architecture by Tseng et al. discussed in Section 4.2, is also capable of computing variable N-point 1D IDCT [30]. Again, specific details are not given. The realignment scheme is mentioned but not described apart from stating that the look-up table outputs reshift the data. Also, the programmable architecture by Chen et al., discussed in Section 4.2, is also capable of variable N-point 1D IDCT [32]. Again, it approximates odd length IDCTs by padding to the next highest even IDCT, and does not address the SA-IDCT realignment operations. The SA-IDCT specific architecture proposed by Hsu et al. has a datapath that uses time-multiplexed adders and multipliers coupled with an auto-aligning transpose memory [37]. It is not clear how their SA-IDCT realignment address generation logic operates. The architecture also employs skipping of all-zero input data to save unnecessary computation, although again specific details discussing how this is achieved are omitted. Hsu et al. admit that the critical path caused by the multipliers in their SA-IDCT architecture limits the maximum operating frequency and has negative power consumption consequences.

5.3. Proposed SA-IDCT architecture

The SA-IDCT architecture proposed in this paper addresses the issues outlined by employing a reconfiguring adder-only structure, similar to the SA-DCT architecture outlined in the previous section. The datapath computes serially each reconstructed pixel \(k (k = 0, \ldots, N - 1)\) of an N-point 1D IDCT by reconfiguring the datapath based on the value of \(k\) and \(N\). Local clock gating is employed using \(k\) and \(N\) to ensure that redundant switching is avoided for power efficiency. For local storage, a TRAM similar to that for the SA-DCT has been designed whose surrounding control logic ensures that the SA-IDCT data realignment is computed efficiently without needless switching or shifting. A pipelined approach alleviates the computational burden of needing to parse the entire shape 8 \(\times\) 8 block before the SA-IDCT can commence.

Due to the additional algorithmic complexity, it is more difficult to design a unified SA-DCT/SA-IDCT module compared to a unified 8 \(\times\) 8 DCT/IDCT module. The reasons for not attempting to do so in the proposed work may be summarised as follows.

1. A video decoder only requires the SA-IDCT. Since SA-DCT and SA-IDCT require different addressing logic, embedding both in the same core will waste area if the final product is a video decoder application only.
2. A video encoder needs both SA-DCT and SA-IDCT, but if real-time constraints are tight, it may be required to have the SA-DCT and SA-IDCT cores executing in parallel. If this is the case, it makes sense to have a ded-
icated task-optimised core. Admittedly, this has negative silicon area implications.

(3) Even though the addressing logic for SA-DCT and SA-IDCT are quite different, the core datapaths that compute the transforms are very similar. Therefore it may be viable to design a unified variable N-point 1D DCT/IDCT datapath and have separate dedicated addressing logic for each. Future work could involve designing such an architecture and comparing its attributes against the two distinct dedicated cores presented in this thesis.

The top-level SA-IDCT architecture is shown in Figure 11, comprising of the TRAM and datapath with their associated control logic. For all modules, local clock gating is employed based on the computation being carried out to avoid wasted power.

5.3.1. Memory and control architecture

The primary feature of the memory and addressing modules in Figure 11 is that they avoid redundant register switching and latency when addressing data and storing intermediate values by manipulating the shape information. The SA-IDCT ACL module parses shape and SA-DCT coefficient data from an external memory and routes the data to the variable N-point 1D IDCT datapath for processing in a row-wise fashion. The intermediate coefficients after the horizontal processing are stored in the TRAM. The ACL then reads each vertical data vector from this TRAM for vertical inverse transformation by the datapath.

Since the alpha information must be fully parsed before any horizontal IDCTs can be computed, the SA-IDCT algorithm requires more computation steps compared to the forward SA-DCT. The proposed ACL tackles this by employing two parallel finite state machines (FSMs) to reduce processing latency—one for alpha parsing and the other for data addressing. As is clear from Figure 12, the parallel FSMs mean that the variable N-point IDCT datapath is continuously fed with data after the first pipeline stage. The shape information is parsed to determine 8 horizontal and 8 vertical $16N$ values, which are stored in a 4-bit register and the shape pattern is stored in a 64-bit register. The shape pattern requires storage for the vertical realignment step, since this realignment cannot be computed from the $N$ values alone.

Once an alpha block has been parsed, the data addressing FSM uses the horizontal $N$ values to read SA-DCT coefficient data from an external memory row by row. Since the shape information is now known, the FSM only reads from memory locations relevant to the VOP. The ACL uses a parallel/interleaved data buffering scheme similar to that for the SA-DCT to maximise throughput. By virtue of the fact that the SA-DCT coefficients are packed into the top left-hand corner of the $8 \times 8$ block, early termination is possible for the horizontal IDCT processing steps. If the horizontal $N$ value for row index $j$ has been parsed as 0, it is guaranteed that
all subsequent rows with index \( j \) will also be 0 since the data is packed. Hence the vertical IDCT processing can begin immediately if this condition is detected.

The data addressing FSM reads intermediate coefficients column-wise from the TRAM for the vertical IDCT processing. Early termination based on \( N = 0 \) detection is not possible in this case since the data is no longer packed. When a column is being read from the TRAM, the data addressing FSM also regenerates the original pixel address from the 64-bit shape pattern. This 64-bit register is divided into an 8-bit register for each column. Using a 3-bit counter, the FSM parses the 8-bit register for the current column until all \( N \) addresses have been found. These addresses are read serially by the \( N \)-point IDCT datapath as the corresponding pixel is reconstructed.

The TRAM is a 64-word \( \times \) 15-bit RAM that stores the reconstructed data produced by the horizontal inverse transform process. This data is then read by the ACL in a transposed fashion and vertically inverse transformed by the datapath yielding the final reconstructed pixels. When storing data here the index \( k \) is manipulated to store the value at address \( 8 \times N_{\text{curr}}[k] + k \). Then \( N_{\text{curr}}[k] \) is incremented by 1. After the entire block has been horizontally inverse transformed, the TRAM has the resultant data packed to the top left corner of the block. For the subsequent vertical inverse transformations, the ACL data addressing FSM combined with the \( N \) value registers inside the TRAM read the appropriate data from the TRAM. Horizontal realignment is intrinsic in the addressing scheme meaning explicit data shifting is not required. Instead, manipulating the shape information combined with some counters control the realignment.

### 5.3.2. Datapath architecture

When loaded, a vector is then passed to the variable \( N \)-point 1D IDCT module (Figure 13), which computes all \( N \) reconstructed pixels serially in a ping-pong fashion (i.e., \( f[N-1], f[0], f[N-2], f[1], \ldots \)). The module is a five-stage pipeline and employs adder-based distributed arithmetic using a multiplexed weight generation module (MWGM) and a partial product summation tree (PPST), followed by even-odd recomposition (EOR). The MWGM and the PPST are very similar in architecture to the corresponding modules used for our SA-DCT described in Section 4.3. From a power consumption perspective, the use of adder-based distributed arithmetic is advantageous since no multipliers are used. The adder tree has been designed in a balanced topology to reduce the probability of glitching. The structure reconfigures according to \([k, N]\) multiplexing the adders appropriately.

The EOR module in Figure 13 exploits the fact that the variable \( N \)-point IDCT matrices are symmetric to reduce the amount of computation necessary, and this necessitates the ping-pong computation order. The EOR module takes successive pairs of samples and recomposes the original pixel values but in a ping-pong order, for example, \( (f(0), f(7), f(1), f(6), \ldots) \) for \( N = 8 \). This data ordering eliminates the need for data buffering that is required if the sequence generated is \( (f(0), f(1), f(2), f(3), \ldots) \). The ping-pong ordering is taken into account by the ACL module responsible for intermediate data storage in the TRAM and vertical realignment of the final coefficients.

### 5.4. Experimental results

Synthesising the design with Synopsys Design Compiler targeting TSMC 0.09 \( \mu \)m TCBN90LP technology yields a gate count of 27518 and a maximum theoretical operating frequency \( f_{\text{max}} \) of 588 MHz. Table 5 shows that the proposed SA-IDCT architecture improves upon the Chen architecture [32] in terms of PGCC by an order of magnitude \((5.2 \times 10^6 \text{ versus } 3.9 \times 10^7)\). Benchmarking against the Hsu architecture [37] is less straightforward since that architecture can operate in zero skipping mode as described in Section 5.2. Also, Hsu et al. do not mention specifically the computational cycle latency of their architecture. They do quote the average throughput in MPixels/sec of their module in both no skip mode and zero skipping mode. From these figures, the authors estimate that the cycle latency in no skip mode is 84.
cycles and averages 10 cycles in zero skipping mode. Compared to the Hsu et al. no skip mode, the proposed architecture, which does not implement zero skipping, improves upon the Hsu architecture in terms of PGCC by an order of magnitude (5.2 × 10^5 versus 3.2 × 10^7). When comparing the proposed architecture against the zero skipping mode of the Hsu architecture, the Hsu architecture is slightly better, although the results have the same order of magnitude (5.2 × 10^6 versus 3.8 × 10^6). However, since the proposed architecture represents an order of magnitude improvement when both are in no skip mode, it is reasonable to assume that in the future if a zero skipping mode is incorporated into the proposed architecture, it will also improve on the zero skipping mode of the Hsu architecture. However, it must be noted that the gate count of the current implementation of the proposed design is much smaller than the Hsu architecture (27 518 versus 37 7685).

The power consumption figure of 0.46 mW was obtained by running back-annotated dynamic simulation of the gate level netlist for various VO sequences and taking an average (@14 MHz, 1.2 V, 25°C). The simulations were run at 14 MHz since this is the lowest possible operating frequency that guarantees 30 fps CIF real-time performance, given a worst case cycle latency per block of 188 cycles. Power consumption results are normalised based on voltage, operating frequency, and technology to give the normalised power (P′), energy (E′), and EDP′ figures in Table 5. Note that the energy figures quoted in the table are the normalised energies required to process a single opaque 8 × 8 block. The proposed SA-IDCT architecture improves upon the Chen architecture in terms of normalised power and energy. Compared to the Hsu architecture (in no skip mode), the proposed architecture again is better in terms of normalised power and energy. Table 5 shows that the Hsu architecture in zero skipping mode outperforms the current implementation of the proposed design (no zero skipping) in terms of energy, despite the fact that the current implementation of the proposed design has a better normalised power consumption performance. This is a direct consequence of the reduced clock cycle latency achievable with a zero skipping scheme. Future work on the proposed SA-IDCT architecture will involve incorporating an appropriate zero skipping scheme. It is expected that this future work will improve the performance of the proposed architecture significantly.

6. CONCLUSIONS

Novel hardware accelerator architectures for the most computationally demanding tools in an MPEG-4 codec have been presented. Using normalised benchmarking metrics, the experimental results presented in this paper show that the proposed architectures improve significantly compared to prior art.

Although the cores presented in this paper are dedicated by nature, they are flexible enough to be reused for multimedia processing tasks other than MPEG-4 compression. Indeed the cores may be considered as “basic enabling technologies” for various multimedia applications. For example, the ME core, albeit configured in a different way, can be used for feature extraction for indexing or depth estimation (when two cameras are available). In terms of future work in this area, we intend to reuse the cores presented in this paper as pre/post processing accelerators for robust face segmentation. Clearly the results of the segmentation can be encoded by MPEG-4 using the same accelerators. Such hardware reuse is also attractive from a low-energy viewpoint.

ACKNOWLEDGMENTS

The support of the Informatics Commercialisation initiative of Enterprise Ireland is gratefully acknowledged. The authors would also like to thank Dr. Valentin Muresan for his significant contributions to this work.

REFERENCES

[1] D. Chai and K. N. Ngan, “Face segmentation using skin-color map in videophone applications,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 4, pp. 551–564, 1999.
[2] MPEG-4: Information Technology—Coding of Audio Visual Objects—Part 2: Visual, ISO/IEC 14496-2, ISO/IEC Std., Rev. Amendment 1, July 2000.

[3] T. Sikora, “The MPEG-4 video standard verification model,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 19–31, 1997.

[4] P.-C. Tseng, Y.-C. Chang, Y.-W. Huang, H.-C. Fang, C.-T. Huang, and L.-G. Chen, “Advances in hardware architectures for image and video coding—a survey,” Proceedings of the IEEE, vol. 93, no. 1, pp. 184–197, 2005.

[5] H.-C. Chang, Y.-C. Wang, M.-Y. Hsu, and L.-G. Chen, “Efficient algorithms and architectures for MPEG-4 object-based video coding,” in Proceedings of IEEE Workshop on Signal Processing Systems (SiPS ’00), pp. 13–22, Lafayette, LA, USA, October 2000.

[6] P. Kuhn, Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation, Kluwer Academic, Dordrecht, The Netherlands, 1st edition, 1999.

[7] P. Landman, “Low-power architectural design methodologies,” Ph.D. dissertation, University of California, Berkeley, Calif, USA, August 1994, http://bwrc.eecs.berkeley.edu/Publications/1994/theses/hw_pwr_arch_des_meth_Landman/Landman94.pdf.

[8] G. K. Yap, Practical Low Power Digital VLSI Design, Kluwer Academic, Dordrecht, The Netherlands, 1st edition, 1998.

[9] A. Kinane, “Energy efficient hardware acceleration of multimedia processing tools,” Ph.D. dissertation, School of Electronic Engineering, Dublin City University, Dublin, Ireland, April 2006, http://www.eeng.dcu.ie/~kinanea/thesis/kinane_final.pdf.

[10] S.-C. Cheng and H.-M. Hang, “A comparison of block-matching algorithms mapped to systolic-array implementation,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 5, pp. 741–757, 1997.

[11] E. Chan and S. Panchanathan, “Motion estimation architecture for video compression,” IEEE Transactions on Consumer Electronics, vol. 39, no. 3, pp. 292–297, 1993.

[12] C.-K. Cheung and L.-M. Po, “Normalized partial distortion search algorithm for block motion estimation,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, no. 3, pp. 417–422, 2000.

[13] Y.-K. Lai and L.-G. Chen, “A data-interlacing architecture with two-dimensional data-reuse for full-search block-matching algorithm,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 2, pp. 124–127, 1998.

[14] Y.-S. Jehng, L.-G. Chen, and T.-D. Chiueh, “An efficient and simple VLSI tree architecture for motion estimation algorithms,” IEEE Transactions on Signal Processing, vol. 41, no. 2, pp. 889–900, 1993.

[15] H. Nakayama, T. Yoshitake, H. Komazaki, et al., “A MPEG-4 video LSI with an error-resilient codec core based on a fast motion estimation algorithm,” in Proceedings of IEEE International Solid-State Circuits Conference (ISSCC ’92), vol. 2, p. 296, San Francisco, Calif, USA, February 2002.

[16] V. L. Do and K. Y. Yun, “A low-power VLSI architecture for full-search block-matching motion estimation,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 4, pp. 393–398, 1998.

[17] M. Takahashi, T. Nishikawa, M. Hamada, et al., “A 60-MHz 240-mW MPEG-4 videophone LSI with 16-Mb embedded DRAM,” IEEE Journal of Solid-State Circuits, vol. 35, no. 11, pp. 1713–1721, 2000.

[18] N. Chang, K. Kim, and H. G. Lee, “Cycle-accurate energy measurement and characterization with a case study of the ARM7TDMI,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 2, pp. 146–154, 2002.

[19] N. Brady, “MPEG-4 standardized methods for the compression of arbitrarily shaped video objects,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 9, no. 8, pp. 1170–1189, 1999.

[20] D. Yu, S. K. Jang, and I. B. Ra, “Fast motion estimation for shape coding in MPEG-4,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 4, pp. 358–363, 2003.

[21] K. Panusopone and X. Chen, “A fast motion estimation method for MPEG-4 arbitrarily shaped objects,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’00), vol. 3, pp. 624–627, Vancouver, BC, Canada, September 2000.

[22] T.-H. Tsai and C.-P. Chen, “A fast binary motion estimation algorithm for MPEG-4 shape coding,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 6, pp. 908–913, 2004.

[23] K.-B. Lee, H.-Y. Chin, N. Y.-C. Chang, H.-J. Hsu, and C.-W. Jen, “A memory-efficient binary motion estimation architecture for MPEG-4 shape coding,” in Proceedings of the 16th European Conference on Circuit Theory and Design (ECCTD ’03), vol. 2, pp. 93–96, Cracow, Poland, September 2003.

[24] K.-B. Lee, H.-Y. Chin, N. Y.-C. Chang, H.-J. Hsu, and C.-W. Jen, “Optimal frame memory and data transfer scheme for MPEG-4 shape coding,” IEEE Transactions on Consumer Electronics, vol. 50, no. 1, pp. 342–348, 2004.

[25] B. Natarajan, V. Bhaskaran, and K. Konstantinides, “Low-complexity block-based motion estimation via one-bit transforms,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 7, no. 4, pp. 702–706, 1997.

[26] T. Larkin, V. Muresan, and N. O’Connor, “A low complexity hardware architecture for motion estimation,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS ’06), pp. 2677–2680, Kos, Greece, May 2006.

[27] M. M. Mizuki, U. Y. Desai, I. Masaki, and A. Chandrakasan, “A binary block matching architecture with reduced power consumption and silicon area requirement,” in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP ’96), vol. 6, pp. 3248–3251, Atlanta, GA, USA, May 1996.

[28] T. Sikora and B. Makai, “Shape-adaptive DCT for generic coding of video,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 5, no. 1, pp. 59–62, 1995.

[29] T. Le and M. Glesner, “Flexible architectures for DCT of variable-length targeting shape-adaptive transforms,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 10, no. 7, pp. 1489–1495, 2000.

[30] P.-C. Tseng, C.-T. Haung, and L.-G. Chen, “Reconfigurable discrete cosine transform processor for object-based video signal processing,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS ’04), vol. 2, pp. 353–356, Vancouver, BC, Canada, May 2004.

[31] K.-H. Chen, J.-I. Guo, J.-S. Wang, C.-W. Yeh, and T.-F. Chen, “A power-aware IP core design for the variable-length DCT/IDCT targeting at MPEG-4 shape-adaptive transforms,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS ’04), vol. 2, pp. 141–144, Vancouver, BC, Canada, May 2004.
[32] K.-H. Chen, J.-I. Guo, J.-S. Wang, C.-W. Yeh, and J.-W. Chen, “An energy-aware IP core design for the variable-length DCT/IDCT targeting at MPEG-4 shape-adaptive transforms,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 5, pp. 704–715, 2005.

[33] K.-B. Lee, H.-C. Hsu, and C.-W. Jen, “A cost-effective MPEG-4 shape-adaptive DCT with auto-aligned transpose memory organization,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS ‘04), vol. 2, pp. 777–780, Vancouver, BC, Canada, May 2004.

[34] A. Kinane, V. Muresan, N. O’Connor, N. Murphy, and S. Marlow, “Energy-efficient hardware architecture for variable N-point 1D DCT,” in Proceedings of International Workshop on Power and Timing Modelling, Optimization and Simulation (PATMOS ’04), pp. 780–788, Santorini, Greece, September 2004.

[35] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, “Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, no. 2, pp. 151–165, 1996.

[36] I. Koren, Computer Arithmetic Algorithms, A. K. Peters, Natick, Mass, USA, 2nd edition, 2002.

[37] H.-C. Hsu, K.-B. Lee, N. Y.-C. Chang, and T.-S. Chang, “An MPEG-4 shape-adaptive inverse DCT with zero skipping and auto-aligned transpose memory,” in Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS ’04), vol. 2, pp. 773–776, Tainan, Taiwan, December 2004.
Research Article
Adaptive Motion Estimation Processor for Autonomous Video Devices

T. Dias, S. Momcilovic, N. Roma, and L. Sousa
INESC-ID/IST/ISEL, Rua Alves Redol 9, 1000-029 Lisboa, Portugal

Received 1 June 2006; Revised 21 November 2006; Accepted 6 March 2007

Recommended by Marco Mattavelli

Motion estimation is the most demanding operation of a video encoder, corresponding to at least 80% of the overall computational cost. As a consequence, with the proliferation of autonomous and portable handheld devices that support digital video coding, data-adaptive motion estimation algorithms have been required to dynamically configure the search pattern not only to avoid unnecessary computations and memory accesses but also to save energy. This paper proposes an application-specific instruction set processor (ASIP) to implement data-adaptive motion estimation algorithms that is characterized by a specialized datapath and a minimum and optimized instruction set. Due to its low-power nature, this architecture is highly suitable to develop motion estimators for portable, mobile, and battery-supplied devices. Based on the proposed architecture and the considered adaptive algorithms, several motion estimators were synthesized both for a Virtex-II Pro XC2VP30 FPGA from Xilinx, integrated within an ML310 development platform, and using a StdCell library based on a 0.18 μm CMOS process. Experimental results show that the proposed architecture is able to estimate motion vectors in real time for QCIF and CIF video sequences with a very low-power consumption. Moreover, it is also able to adapt the operation to the available energy level in runtime. By adjusting the search pattern and setting up a more convenient operating frequency, it can change the power consumption in the interval between 1.6 mW and 15 mW.

Copyright © 2007 T. Dias et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Motion estimation (ME) is one of the most important operations in video encoding to exploit temporal redundancies in sequences of images. However, it is also the most computationally costly part of a video codec. Despite that, most of the actual video coding standards apply the block matching (BM) ME technique on reference blocks and search areas of variable size [1]. Nevertheless, although the BM approach simplifies the ME operation by considering the same translation movement for the whole block, real-time ME with power consumption constraints is usually only achievable with specialized VLSI processors [2]. In fact, depending on the adopted search algorithm, up to 80% of the operations required to implement a MPEG-4 video encoder are devoted to ME, even when large search ranges are not considered [3].

The full-search block-matching (FSBM) [4] method has been, for several years, the most adopted method to develop VLSI motion estimators, due to its regularity and data independency. In the 90s, several nonoptimal but faster search block-matching algorithms were proposed, such as the three-step search (3SS) [5], the four-step search (4SS) [6], and the diamond search (DS) [7]. However, these algorithms have been mainly applied in pure software implementations, better suited to support data-dependency and irregular search patterns, which usually result in complex and inefficient hardware designs, with high power consumption.

The recent demand for the development of portable and autonomous communication and personal assistant devices imposes additional requirements and constraints to encode video in real time but with low power consumption, maintaining a high signal-to-noise ratio for a given bitrate. Recently, the FSBM method was adapted to design low-power architectures based on a ±1 full-search engine that implements a fixed 3 × 3 square search window [8], by exploiting the variations of input data to dynamically configure the search-window size [9], or to guide the search pattern according to the gradient-descent direction.

Moreover, new data-adaptive efficient algorithms have also been proposed, but up until now only software implementations have been presented. These algorithms avoid unnecessary computations and memory accesses by taking...
advantage of temporal and spacial correlations, in order to adapt and optimize the search patterns. These are the cases of the motion vector field adaptive search technique (MV-FAST), the enhanced predictive zonal search (EPZS) [10, 11], and the fast adaptive motion estimation (FAME) [12]. In these algorithms, the correlations are exploited by carrying information about previously computed MVs and error values, in order to predict and adapt the current search space, namely, the start search location, the search pattern, and the search area size. These algorithms also comprise a limited number of different states. Such states are selected according to threshold values that are dynamically adjusted to adapt the search procedure to the video sequence characteristics.

This paper proposes a new architecture and techniques to implement efficient ME processors with low power consumption. The proposed application-specific instruction set processor (ASIP) platform was tailored to efficiently program and implement a broad class of powerful, fast and adaptive ME search algorithms, using both the traditional fixed block structure (16 × 16 pixels), adopted in H.261/H.263 and MPEG-1/MPEG-2 video coding standards, or even variable-size, block structures, adopted in H.264/AVC coding standards. Such flexibility was attained by developing a simple and efficient microarchitecture to support a minimum and specialized instruction set, composed by only eight different instructions specifically defined for ME. In the core of this architecture, a datapath has been specially designed around a low-power arithmetic unit that efficiently computes the sum of absolute differences (SAD) function. Furthermore, the several control signals are generated by a quite simple and hardwired control unit.

A set of software tools was also developed and made available to program ME algorithms on the proposed ASIP, namely, an assembler and a cycle-based accurate simulator. Efficient and adaptive ME algorithms, that also take into account the amount of energy available in portable devices at any given time instant, have been implemented and simulated for the proposed ASIP. The proposed architecture was described in VHDL and synthesized for a Virtex-II Pro FPGA from Xilinx. An application-specific integrated circuit (ASIC) was also designed, using a 0.18 μm CMOS process. Experimental results show that the proposed ASIP is able to encode video sequences in real time with very low power consumption.

This paper is organized as follows. In Section 2, ME algorithms are described and adaptive techniques are discussed. Section 3 presents the instruction set and the microarchitecture of the proposed ASIP. Section 4 describes the software tools that were developed to program and simulate the operation of the ASIP with cycle level accuracy, as well as other implementation aspects. Experimental results are provided in Section 5, where the efficiency of the proposed ASIP is compared with the efficiency of other motion estimators. Finally, Section 6 concludes the paper.

2. ADAPTIVE MOTION ESTIMATION

Block matching algorithms (BMA) try to find the best match for each macroblock (MB) in a reference frame, according to a search algorithm and a given distortion measure. Several search algorithms have been proposed in the last few years, most of them using the SAD distortion measure, depicted in (1), where \( F_{\text{curr}} \) and \( F_{\text{prev}} \) denote the current and previously coded frames, respectively,

\[
SAD(v_x, v_y) = \sum_{m=0}^{(N_x-1)} \sum_{n=0}^{(N_y-1)} |F_{\text{curr}}(x + m, y + n) - F_{\text{prev}}(x + v_x + m, y + v_y + n)|.
\]

The well-known FSBM algorithm examines all possible displaced candidates within the search area, providing the optimal solution at the cost of a huge amount of computations. The faster BMAs reduce the search space by guiding the search pattern according to general characteristics of the motion, as well as the computed values for distortion. These algorithms can be grouped in two main classes: (i) algorithms that treat each macroblock independently and search according to predefined patterns, assuming that distortion decreases monotonically as the search moves in the best match direction; (ii) algorithms that also exploit interblock correlation to adapt the search patterns.

The 3SS, 4SS, and DS are well-known examples of fast BMAs that use a square search pattern. Their main advantage is their simplicity, being the \( a \) priori known possible sequence of locations that can be used in the search procedure. The 3SS algorithm examines nine distinctive locations at \( 9 \times 9, 5 \times 5, \) and \( 3 \times 3 \) pixel search windows. In 4SS, search windows have \( 5 \times 5 \) pixels in the first three steps and \( 3 \times 3 \) pixels in the fourth step. If the minimal distortion point corresponds to the center in any of the intermediate steps, this algorithm goes directly to the fourth and last step. On the other hand, the DS algorithm performs the search within the limits of the search area until the best matching is found in the center of the search pattern. It applies two diamond-shaped patterns: large diamond search pattern (LDSP), with 9 search points, and small diamond search pattern (SDSP) with 5 search points. The algorithm initially performs the LDSP, moving it in the direction of a minimal distortion point until it is found in the center of a large diamond. After that, the SDSP is performed as a final step.

The other class of more powerful and adaptive fast BMAs exploits interblock correlation, which can be in both the space and time dimensions. With this approach, information from adjacent MBs is potentially used to obtain a first prediction of the motion vector (MV). The MVFAST and the FAME are some examples of these algorithms.

The MVFAST is based on the DS algorithm, by adopting both the LDSP and the SDSP along the search procedure (see Figure 1). The initial central search point as well as the following search patterns are predicted using a set of adjacent MBs, namely, the left, the top, and the top-right neighbor MBs depicted in Figure 1(a). The selection between LDSP and SDSP is performed based on the characteristics of the motion in the considered neighbor MBs and on the values of two thresholds, L1 and L2. As a consequence, the algorithm performs as follows: (i) when the magnitude of the largest
MV of the three neighbor MBs is below a given threshold $L_1$, the algorithm adopts an SDSP, starting from the center of the search area and moving the small diamond until the minimum distortion is found in the center of the diamond; (ii) when the largest MV is between $L_1$ and $L_2$, the algorithm uses the same central point but applies the LDSP until the minimal distortion block is found in the center; an additional step is performed with the SDSP; (iii) when the magnitude is greater than $L_2$, the minimum distortion point among the predictor MVs is chosen as the central point and the algorithm performs the SDSP until the minimum distortion point is found in the center.

Meanwhile, the predictive motion vector field adaptive search technique (PMVFAST) algorithm has been proposed. It incorporates a set of thresholds in the MVFAST to trade higher speedup at the cost of memory size and memory bandwidth. It computes the SAD of some highly probable MVs and stops if the minimum SAD so far satisfies the stopping criterion, performing a local search using some of the techniques of MVFAST.

More recently, the FAME [12] algorithm was proposed, claiming very accurate MVs that lead to a quality level very close to the FSBM but with a significant speedup. The FAME algorithm outperforms MVFAST by taking advantage of the correlation between MVs in both the spatial (see Figures 2(b)–2(d)) and the temporal (see Figure 2(e)) domains, using adaptive thresholds and adaptive diamond-shape search patterns to accelerate ME. To accomplish such objective, it features an improved control to confine the search pattern and avoid stationary regions.

When compared in terms of computational complexity, all these algorithms are widely regarded as good candidates for software implementations, due to their inherent irregular processing nature. It is proved in this paper that by adopting the proposed ASIP approach, it is possible to develop hardware processors to efficiently implement not only any adaptive ME algorithm of this class, but also any other fast BMA. In fact, the FSBM, 3SS, 4SS, DS, MVFAST, and FAME algorithms have been implemented with the proposed ASIP, in order to evaluate the performance of the processor.

3. ASIP INSTRUCTION SET AND MICROARCHITECTURE

3.1. Instruction set

The instruction set architecture (ISA) of the proposed ASIP was designed to meet the requirements of most ME algorithms, including adaptive ones, but optimized for portable and mobile platforms, where power consumption and implementation area are mandatory constraints. Consequently, such ISA is based on a register-register architecture and provides only a reduced number of different operations (eight) that focus on the most widely executed instructions in ME algorithms. This register-register approach was adopted due to its simplicity and efficiency, allowing the design of simpler and less hardware consuming circuits. On the other hand, it offers increased efficiency due to its large number of general purpose registers (GPRs), which provides a reduction of the memory traffic and consequently a decrease in the program execution time. The amount of registers that compose the register file therefore results as a tradeoff between the implementation area, memory traffic, and the size of the program memory. For the proposed ASIP, the register file consists of...
24 GPRs and eight special purpose registers (SPRs) capable of storing one 16 bits word each. Such configuration optimizes the ASIP efficiency since: (i) the amount of GPRs is enough to allow the development of efficient, yet simple, programs for most ME algorithms; (ii) the 16 bits data type covers all the possible values assigned to variables in ME algorithms; and (iii) the eight SPRs are efficiently used as configuration parameters for the implemented ME algorithms and for data I/O.

The operations supported by the proposed ISA are grouped in four different categories of instructions, as can be seen from Table 1, and were obtained as the result of the analysis of the execution of several different ME algorithms [10, 11, 13]. The encoding of the instructions into binary representation was performed using 16 bits and a fixed format. For each instruction it is specified an opcode and up to three operands, depending on the instruction category. Such encoding scheme therefore provides minimum bit wasting for instruction encoding and eases the decoding, thus allowing a good tradeoff between the program size and the efficiency of the architecture. In the following, it is presented a brief description of all the operations of the proposed ISA.

### 3.1.1. Control operation

The jump control operation, J, introduces a change in the control flow of a program, by updating the program counter with an immediate value that corresponds to an effective address. The instruction has a 2 bits condition field (cc) that specifies the condition that must be verified for the jump to be taken: always or in case the outcome of the last executed arithmetic or graphics operation (SAD16) is negative, positive or zero. Not only is this instruction important for algorithmic purposes, but also for improving code density, since it allows a minimization of the number of instructions required to implement an ME algorithm and therefore a reduction of the required capacity of the program memory.

### 3.1.2. Register data transfer operations

The register data transfer operations allow the loading of data into a GPR or SPR of the register file. Such data can be the content of another register in the case of a simple move instruction, MOVR, or an immediate value for constant loading, MOVC. Due to the adopted instruction coding format, the immediate value is only 8 bits width, but a control field (t) sets the loading of the 8 bits literal into the destination register upper or lower byte.

### 3.1.3. Arithmetic operations

In what concerns the arithmetic operations, while the ADD and SUB instructions support the computation of the coordinates of the MBs and of the candidate blocks, as well as the updating of control variables in loops, the DIV2 instruction (integer division by two) allows, for example, to dynamically adjust the search area size, which is most useful in adaptive ME algorithms. Moreover, these three instructions also provide some extra information about its outcome that can be used by the jump (J) instruction, to conditionally change the control flow of a program.

### 3.1.4. Graphics operation

The SAD16 operation allows the computation of the SAD similarity measure between an MB and a candidate block. To do so, this operation computes the SAD value considering two sets of sixteen pixels (the minimum amount of pixels for an MB in the MPEG-4 video coding standard) and accumulates the result to the contents of a GPRs. The computation of a SAD value for a given (16 × 16)-pixel candidate MB therefore requires the execution of sixteen consecutive SAD16 operations. To further improve the algorithm efficiency and reduce the program size, both the horizontal and vertical coordinates of the line of pixels of the candidate block under processing are also updated with the execution of this operation. Likewise the arithmetic operations, the outcome of this operation also provides some extra information that can be used by the jump (J) instruction to conditionally change the control flow of a program.

### 3.1.5. Memory data transfer operation

The processor comprises two small and fast local memories, to store the pixels of the MB under processing and of its corresponding search area. To improve the processor performance, a memory data transfer operation (LD) was also included, to load the pixel data into these memories. Such
operation is carried out by means of an address generation unit (AGU), which generates the set of addresses of both the corresponding internal memory as well as of the external frame memory, that are required to transfer the pixel data. The selection of the target memory is carried out by means of a 1-bit control field, which is used to specify the type of image area that is loaded into the local memory. As a consequence, this operation is performed independently for the data concerning a given MB and for the corresponding search area.

### 3.2. Micro architecture

The proposed ISA is supported by a specially designed micro-architecture, following strict power and area driven policies to support its implementation in portable and mobile platforms. This micro-architecture presents a modular structure and is composed by simple and efficient units to optimize the data processing, as it can be seen from Figure 3.

#### 3.2.1. Control unit

The control unit is characterized by its low complexity, due to the adopted fixed instruction encoding format and a careful selection of the opcodes for each instruction. This not only provided the implementation of a very simple and fast hardwired decoding unit, which enables almost all instructions to complete in just one clock cycle, but also allowed the implementation of effective power saving policies within the processors functional units, such as clock gating and operating frequency adjustment. The former technique is applied to control the switching activity at the function unit level, by inhibiting input updates to functional units whose outputs are not required for a given operation, while the latter adjusts the operating frequency according to the programmed algorithm and the current available energy level.

#### 3.2.2. Datapath

For more complex and specific operations, like the LD and SAD16 instructions, the datapath also includes specialized units to improve the efficiency of such operations: the AGU and the SAD unit (SADU), respectively.

The LD operation is executed by a dedicated AGU optimized for ME, which is capable of fetching all the pixels for both an MB and an entire search area. To maximize the efficiency of the data processing, this unit can work in parallel with the remaining functional units of the micro-architecture. Using such feature, programs can be optimized by rescheduling the LD instructions to allow data fetching from memory to occur simultaneously with the execution of other parts of the program that do not depend on this data. For implementations imposing strict constraints in the power consumption, memory accesses can be further optimized by using efficient data reuse algorithms and extra hardware structures [4, 14]. This not only significantly reduces the memory traffic to the external memory, but also provides a considerable reduction in the power consumption of the video encoding system.

The SADU can execute the SAD16 operation in up to sixteen clock cycles and is capable of using the arithmetic and logic unit (ALU) to update the coordinates of the candidate block line of pixels. The number of clock cycles required for the computation of a SAD value is imposed by the type of architecture adopted to implement this unit, which depends on the power consumption and implementation area constraints specified at design time. Thus, applications imposing more severe constraints in power or area can use a serial processing architecture, that reuses hardware but takes more clock cycles to compute the SAD value, while others without so strict requisites may use a parallel processing architecture that is able to compute the SAD value in only one single clock cycle. Pipelined versions of the SADU are also supported to allow better tradeoffs between the latency, the power consumption, and the required implementation area, thus providing increased flexibility for different implementations of the proposed ASIP.

Despite all these different alternatives in what concerns the SADU architecture to meet the desired performance level, the implemented SADU also adopted an innovative and efficient arithmetic unit to compute the minimum SAD distance [15] that allows the proposed processor to better comply with the low-power constraints usually found in autonomous and portable handheld devices. Such unit not only avoids the usage of carry-propagate cells to compute and compare the SAD metric, by adopting carry-free arithmetic,
but it also generates a "greater or equal" (GE) signal, issued by the best-match detection unit (see Figure 4). This signal is obtained from the partial values of the SAD measure, by comparing the current metric value with the best one previously obtained. It can be used by the main state machine to update the output register corresponding to the current MV. Due to its null latency, this GE signal can also be used to apply the set of power-saving techniques that have been proposed in the last few years [16]. In fact, it is used as a control mechanism to avoid needless calculations in the computation of the best match for a macroblock, by aborting the ME procedure as soon as the partial value of the distance metric for the candidate block under processing exceeds the one already computed for the current block [16]. Such computations can be avoided by disabling all the logic and arithmetic units used in the computation of the SAD metric, thus providing significant power saving ratios. On average, this technique allows to avoid up to 50% of the required computations [16], giving rise to a reduction of up to 75% of the overall power consumption [15].

3.2.3. External interface

The proposed ASIP presents an external interface with a quite reduced pin count, as shown in Figure 5, that allows an easy embedding of the presented micro-architecture in both existing and future video encoders. Such interface was designed not only to allow efficient data transfers from the external frame memory, but also to efficiently export the coordinates of the best matching MVs to the video encoder. In addition, it also provides the possibility to download the processor’s firmware, that is, the compiled assembly code of the desired ME algorithm.

Since pixels for ME are usually represented using 8 bits and MVs are estimated using pixels from the current and previous frames (each frame consists of $704 \times 576$ pixels in the 4CIF image format), the interface with the external frame memory was designed to allow 8 bits data transfers from a 1 MB memory address space. Thus, the proposed interface with such external memory bank is done using three I/O ports: (i) a 20 bits output port that specifies the memory address for the data transfers ($addr$); (ii) an 8 bits bidirectional port for transferring the data ($data$); and (iii) a 1-bit output port that sets whether it is a load or store operation ($#oe\_we$). Since the external frame memory is to be shared between the video encoder and the ME circuit, the proposed ASIP interface has two extra 1-bit control ports to implement the required handshake protocol with the bus master: the $req$ port allows requesting control of the bus to the bus master, while the $gnt$ port allows the bus master to grant such control.

To minimize the number of required I/O connections, the coordinates of the best matching MVs are also outputted through the $data$ port. Nevertheless, such operation requires two distinct clock cycles for its completion: a first one to output the low-order 8 bits of the MV coordinate and a second one to output its high-order 8 bits. In addition, every time a new value is outputted through the $data$ port, the status of the $done$ output port is toggled, in order to signal the video encoder that new data awaits to be read at the $data$ port.

This port is also used to dynamically acquire the energy level that is available to compute the motion estimation at any instant (see Figure 5). Such level may be used by adaptive algorithms to adjust the overall computational cost of the ME procedure.

The processor’s firmware, corresponding to the compiled assembly code of the considered ME algorithm, is also downloaded into the program RAM through the $data$ port. To do so, the processor must be in the programming mode, which it enters whenever a high level is simultaneously set into the $rst$ and $en$ input ports. In this operating mode, after having acquired the bus ownership, the master processor supplies memory addresses through the $addr$ port and loads the corresponding instructions into the internal program RAM. The
... 
0042h 81eh SAD16 R1, R14, R15
0043h 81eh SAD16 R1, R14, R15
0044h 81eh SAD16 R1, R14, R15
0045h 81eh SAD16 R1, R14, R15
0046h eb21h SUB R11, R2, R1
0047h 684bh J.N NEXT_POS
0048h 2041h MOVR R2, R1
0049h 20c5h MOVR R6, R5
004ah 204eh MOVR R7, R4
004bh NEXT_POS:
004bh c448h ADD R4, R4, R8
004ch eb94h SUB R11, R9, R4
004dh 6850h J.Z NEWLINE
004eh c338h ADD R3, R3, R8
004fh 680eh J.U LOOP
0050h NEWLINE:
0050h 2091h MOVR R4, R17
0051h e044h SUB R4, R0, R4
0052h c558h ADD R5, R5, R8
0053h eb95h SUB R11, R9, R5
0054h 6858h J.Z END
0055h 2170h MOVR R11, R16
0056h c338h ADD R3, R3, R11
0057h 680eh J.U LOOP
0058h END: 
... 

Figure 6: Fraction of one of the output files obtained with the assembly compiler.

processor exits this programming mode as soon as the last memory position of the 1 kB program memory is filled in. Once again, each 16 bits instruction takes two clock cycles to be loaded into the program memory, which is organized in the little-endian format.

4. SOFTWARE TOOLS

To support the development and implementation of ME algorithms using the proposed ASIP, a set of software tools was developed and made available, namely, an assembly compiler and a cycle-based accurate simulator.

Since the proposed ASIP architecture and the considered instruction set do not support subroutine calls nor make use of an instruction/data stack, the implementation of the compiler consists of a straightforward parsing of the assembly instruction directives (as well as their register operands), followed by a corresponding translation into the appropriate opcodes, in order to translate the sequence of assembly instructions into a series of 16 bits machine code words of program data. The exception to this direct translation occurs whenever a jump instruction has to be compiled. A two-step strategy was adopted to compile these control flow instructions, in order to determine the target address of each jump invoked within the program.

In Figure 6 it is presented a fraction of one of the output files (code.lst) that are generated during this translation process. This file presents three different sorts of information, disposed in three columns (see Figure 6). While the first column presents the effective address of each instruction (or label), the second column presents the instruction code of the assembly directive presented in the third column. In the illustrated case, it is presented a fraction of an implementation of the FSBM algorithm (used as reference in the considered algorithm comparisons). As it can be seen in Figure 6, the resulting SAD value, accumulated in R1 register after a sequence of 16 SAD16 instructions (one for each row of the macroblock), is compared with the best SAD value (stored in R2) that was found in previous computations. Depending on the difference between these values, the current SAD value, as well as the corresponding MV coordinates (R5, R4), will be stored in R2, R6, and R7 registers, in order to be considered in the next searching locations. In the remaining instructions, the MV coordinates are incremented and it is checked if the last column and line of the considered search area were already reached, respectively.

The implementation and evaluation of the ME algorithms were supported by implementing an accurate simulator of the proposed ASIP. It provides important information about: the number of clock cycles required to carry out the ME of a given macroblock, the amount of memory space required to store the program code, the obtained motion vector and corresponding SAD value, and so forth.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

To assess the performance provided by the proposed ASIP, the microarchitecture core was implemented by using the described serial processing architecture for the SADU module (see Figure 4) and a simplified AGU that does not allow data reusage. This microarchitecture was described using both behavioral and fully structural parameterizable IEEE-VHDL. The ASIP was firstly implemented in a FPGA device, in order to proof the concept. Later, an ASIC was specifically designed in order to evaluate the efficiency of the proposed architecture and of the corresponding ISA for motion estimation.

The performance of the proposed ASIP was evaluated by implementing several ME algorithms, such as the FSBM, the 4SS, the DS, and the MVFAST and FAME adaptive ME algorithms. These algorithms were programmed with the proposed instruction set and the ASIP operation was simulated by using the developed software tools (see Section 4). Such simulation phase was fundamental to obtain the number of clock cycles required to implement the algorithms, which implicitly defines the minimum clock frequency for real-time processing, as well as the size of the memory required to store the programs.

Table 2 provides the average number of clock cycles per pixel (CPP) required to implement the several considered algorithms, using the following benchmark video sequences: mobile, carphone, foreman, table tennis, bus, and bream. These are well-known video sequences with quite different characteristics, in terms of both movement and spacial detail. The presented results were obtained for a search area with 16 × 16 candidate locations and for the first 20 frames of each video
Table 2: Required clock cycles to process each pixel considering several different algorithms and video sequences.

| Video seq. | FSBM | 4SS | DS | MVFAST | FAME |
|------------|------|-----|----|--------|------|
| Mobile     | 265  | 19  | 15 | 9      | 8    |
| Carphone   | 265  | 21  | 18 | 13     | 9    |
| Foreman    | 265  | 21  | 18 | 13     | 9    |
| Table tennis | 265 | 19  | 15 | 8      | 6    |
| Bream      | 265  | 19  | 15 | 8      | 6    |
| Bus        | 265  | 24  | 21 | 18     | 8    |
| Maximum    | 265  | 24  | 21 | 18     | 9    |

Table 3: Required operating frequencies to process QCIF and CIF video sequences in real time.

| Format | FSBM | 4SS | DS | MVFAST | FAME |
|--------|------|-----|----|--------|------|
| QCIF   | 800 MHz | 75 MHz | 65 MHz | 55 MHz | 28 MHz |
| CIF    | 200 MHz | 20 MHz | 18 MHz | 13 MHz | 8 MHz  |

Table 4: Code size of the proposed algorithms (words of 16 bits).

| Algorithm | FSBM | 4SS | DS | MVFAST | FAME |
|-----------|------|-----|----|--------|------|
| Code size | 56   | 365 | 460 | 744    | 917  |

Table 5: Experimental results of the implementation of the video encoder in the Xilinx ML310 development board.

| Unit          | Slices | LUTs | BRAMs | F (MHz) |
|---------------|--------|------|-------|---------|
| AMEP core     | 2052   | 14%  | 2289  | 8%      | 208    | 67.18  |
| Interface     | 207    | 1%   | 382   | 1%      | 0      | 156.20 |

sequence. Moreover, redundancy was eliminated in both the 4SS and the MVFAST algorithms, by avoiding the computation of SAD more than once for a single location.

The results presented in Table 2 evidence the huge reduction of the number of performed computations that can be achieved when fast search algorithms are applied. The MVFAST and FAME adaptive algorithms allow to significantly reduce the CPP even further, when compared with the 4SS and the DS fast algorithms. By considering the maximum value for the obtained CPPs ($CPP_M$) and a real-time frame rate of 30 Hz for an $H \times W$ image format, the required minimum operating frequency ($\phi$) can be calculated for each class of algorithms using (2),

$$\phi = (H \times W) \times CPP_M \times 30 \text{ Hz.}$$ (2)

By considering the quarter common intermediate format (QCIF) and the common intermediate format (CIF) image formats, as well as the values presented in Table 2 and (2), the required minimum clock frequencies were computed and are presented in Table 3. The obtained operating frequencies of the proposed motion estimators for fast adaptive search algorithms are significantly lower than the operating frequency of the ± 1 full-search-based processor presented in [8].

In Table 4, it is represented the size of the memory required to store the programs corresponding to the considered algorithms. As it can be seen, the adaptive algorithms require significantly more memory for storing the program than the 4SS. The memory requirements of the FAME algorithm are even greater than the MVFAST, due to the need to keep in memory more past information to achieve significantly better predictions. In fact, it requires approximately 13 times more memory than the FSBM. This is the price to pay for the irregularity and also for the adaptability of the MVFAST and FAME algorithms ($744 \times 16$ bit). However, since most of the portable communication systems already provide nonvolatile memories with significant capacity, the power consumption gain due to the reduction of the operating frequency can supersede this disadvantage.

5.1. FPGA implementation for proof of concept

To validate the functionality of the proposed ASIP in a practical realization, a hybrid video encoder was developed and implemented in a Xilinx ML310 development platform, making use of a Virtex-II Pro XC2VP30 FPGA device from Xilinx embedded in the board [17]. Besides all the implementation capabilities offered by such configurable device, this platform also provides two Power-PC processors, several block RAMs (BRAMs), and high speed on-chip bus communication links to enable the interconnection of the Power-PC processors with the developed hardware circuits.

The prototyping video encoding system was implemented by using these resources. It consists of the developed ASIP motion estimator, a software implementation of an H.263 video encoder, built into the FPGA BRAMs and running on a 100–300 MHz Power-PC 405D5 processor, and of four BRAMs to implement the firmware RAM and the local memory banks in the AGU of the proposed ASIP. Furthermore, the Power-PC processor and the developed motion estimator were interconnected according to the interface scheme described in Figure 5, using both the high-speed 64 bits processor local bus (PLB) and the general purpose 32 bits on-chip peripheral bus (OPB), where the Power-PC was connected as the master device. Such interconnect buses are used not only to exchange the control signals between the Power-PC and the proposed ASIP, but also to send all the required data to the proposed motion estimator, namely, the ME algorithm program code and the pixels for both the candidate and reference blocks. Moreover, a simple handshake protocol is used in these data transfers to bridge the different operating frequencies of the two processors.

The operating principle of the proposed prototyping hybrid video encoder consists only of three different tasks related to motion estimation: (i) configuration of the ME co-processor, by downloading an ME algorithm and all the configuration parameters (MB size, search area size, image width, and image height) into the code memory and the SPRs of the proposed ASIP; (ii) data transfers from the Power-PC to the proposed ASIP, which occur on demand by the motion estimator and are used either to download the MB and the search area pixels into the AGU local memories or to supply additional information required by adaptive ME algorithms, depending on the memory position addressed by
the ASIP; and (iii) data transfers from the proposed ASIP to the Power-PC, that are used to output the coordinates of the best-match MV and the corresponding SAD value, as well as the current configuration parameters of the motion estimator, since some adaptive ME algorithms change these values during the video coding procedure.

Table 5 presents the experimental results that were obtained with the implementation of the proposed video coding system in the Virtex-II Pro XC2VP30 FPGA device. Such results show that by using the proposed ASIP, it is possible to estimate MVs in real time (30 fps) for the QCIF and CIF image formats by using any fast or adaptive search algorithms, except the 4SS for CIF images (see Table 3). Moreover, the minimum throughput achieved for the considered algorithms (4SS) is about 2.8 Mpxels/s, corresponding to a relative throughput per slice of about 1.36 kpxels/s/slice.

The operating frequency of the ASIP can be changed in the FPGA by using the digital clock managers (DCMs). In this case, the DCMs were used to configure setup pairs of algorithms/formats-frequencies depicted in Table 3. However, in an ASIC implementation, an additional input is required in the ASIP in order to sense, at any time, the amount of energy that is still available; and an extra programmable divisor to adjust the clock frequency. The control of this dynamic adjustment can be done by the ASIP and the programming of the divisor can be done through an extra output register.

### 5.2. Standard-cell implementation

The proposed motion estimator was implemented using the Synopsis synthesis tools and a high-performance StdCell library based on a 0.18 μm CMOS process from UMC [18]. The obtained experimental results concern an operating environment imposing typical operating conditions: \( T = 25 \, ^{\circ}\text{C}, \) \( V_{dd} = 1.8 \, \text{V}, \) the “suggested 20 k” wire load model, and some constraints that lead to an implementation with minimum area. Typical case conditions have been considered for power estimation, and prelayout netlist power dissipation results are presented.

The first main conclusion that can be drawn from the synthesis results presented in Tables 6, 7, and 8 is that the power consumption of the proposed ASIP for ME with the adaptive ME algorithms is very low. Operating at a frequency of 8 MHz, it only consumes about 1.6 mW, which does not imply any significant reduction of the life time of our actual batteries (typically 1500 mAh batteries). For the 4SS algorithm, the operating frequency increases to about 20 MHz but the power consumption is kept low, about 3.9 mW. The setup corresponding to the FSBM algorithm for the CIF image format was not fully synthesized, since the required operating frequency is beyond the technology capabilities. The maximum operating frequency obtained with this architecture and with this technology is about 144 MHz, as it can be seen in Table 6. Near this maximum frequency, which corresponds to having the components of the processor operating at 100 MHz, the power consumption becomes approximately 20 mW (see Table 7).

Tables 7 and 8 present the power consumption values estimated for the required minimum operating frequencies. Two main clusters of points can be identified in the plot of Figure 7: the one for the QCIF and the one for the CIF format. The former format requires operating frequencies below 25 MHz and the corresponding power consumption is below 6 mW, while for the CIF format the operating frequency is above 50 MHz and the power consumption is between 10 mW and 15 mW. The exception is the FAME algorithm, for which the operating frequency (28 MHz) and the power consumption (5.5 mW) values for the CIF format are closer to the QCIF values.

Common figures of merit for evaluating the energy and the area efficiencies of the video encoders are the number of Mpxels/s/W and the number of Mpxels/s/mm². For the designed VHDL motion estimator, the efficiency figures are, on average, 23.7 Mpxels/s/mm² and 544 Mpxels/s/W. These
values can be compared with the ones that were presented for the motion estimator ASIP proposed in [19], after normalizing the power consumption values to a common voltage level: 22 Mpixels/s/mm² and 323 Mpixels/s/W. Hence, it can be concluded that the proposed motion estimator is more efficient in terms of both power consumption and implementation area. In fact, the improvements should be even greater, since the proposed circuit was designed with a 0.18 μm CMOS technology, while the circuit in [19] was designed with a 0.13 μm CMOS technology.

6. CONCLUSIONS

An innovative design flow to implement efficient motion estimators was presented here. Such approach is based on an ASIP platform, characterized by a specialized datapath and a minimum and optimized instruction set, that was specially developed to allow an efficient implementation of data-adaptive ME algorithms. Moreover, it was also presented a set of software tools that were developed and made available, namely, an assembler compiler and a cycle-based accurate simulator, to support the implementation of ME algorithms using the proposed ASIP.

The performance of the proposed ASIP was evaluated by implementing a hybrid video encoder using regular (FSBM), irregular (4SS and DS), and adaptive (MVFAST and FAME) ME algorithms using the developed software tools and a Xilinx ML310 prototyping environment, that includes a Virtex-II Pro XC2VP30 FPGA. In a later stage, the performance of the developed microarchitecture was also assessed by synthesizing it for an ASIC using a high-performance StdCell library based on a 0.18 μm CMOS process.

The presented experimental results proved that the proposed ASIP is capable of estimating MVs in real time for the QCIF image format for all the tested fast ME algorithms, running at relatively low operating frequencies. Furthermore, the results also showed that the power consumption of the proposed architecture is very low: near 1.6 mW for the adaptive FAME algorithm and around 4 mW for the remaining irregular algorithms that were considered. Consequently, it can be concluded that the low-power nature of the proposed architecture and its high performance make it highly suitable for implementations in portable, mobile, and battery-supplied devices.

REFERENCES

[1] F. C. N. Pereira and T. Ebrahimi, The MPEG4 Book, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.
[2] V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards: Algorithms and Architectures, Kluwer Academic Publishers, Boston, Mass, USA, 2nd edition, 1997.
[3] P. Firsch, N. Demasieux, and W. Gehrke, “VLSI architectures for video compression—a survey,” Proceedings of the IEEE, vol. 83, no. 2, pp. 220–246, 1995.
[4] T. Dias, N. Roma, and L. Sousa, “Efficient motion vector refinement architecture for sub-pixel motion estimation systems,” in Proceedings of IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS ’05), pp. 313–318, Athens, Greece, November 2005.
[5] R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, no. 4, pp. 438–442, 1994.
[6] L.-M. Po and W.-C. Ma, “A novel four-step search algorithm for fast block motion estimation,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, pp. 313–317, 1996.
[7] S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast block-matching motion estimation,” IEEE Transactions on Image Processing, vol. 9, no. 2, pp. 287–290, 2000.
[8] S.-Y. Huang and W.-C. Tsai, “A simple and efficient block motion estimation algorithm based on full-search array architecture,” Signal Processing: Image Communication, vol. 19, no. 10, pp. 975–992, 2004.
[9] S. Saponara and L. Fanucci, “Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems,” IEEE Proceedings Computers & Digital Techniques, vol. 151, no. 1, pp. 51–59, 2004.
[10] A. M. Tourapis, O. C. Au, and M. L. Liou, “Predictive motion vector field adaptive search technique (PMVFAST): enhancing block-based motion estimation,” in Proceedings of Visual Communications and Image Processing (VCIP ’01), vol. 4310 of Proceedings of SPIE, pp. 883–892, San Jose, Calif, USA, January 2001.
[11] A. M. Tourapis, “Enhanced predictive zonal search for single and multiple frame motion estimation,” in Proceedings of Visual Communications and Image Processing (VCIP ’02), vol. 4671 of Proceedings of SPIE, pp. 1069–1079, San Jose, Calif, USA, January 2002.
[12] I. Ahmad, W. Zheng, J. Luo, and M. Liu, “A fast adaptive motion estimation algorithm,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 3, pp. 420–438, 2006.
[13] S. Momčilović, T. Dias, N. Roma, and L. Sousa, “Application specific instruction set processor for adaptive video motion estimation,” in Proceedings of the 9th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD ’06), pp. 160–167, Dubrovnik, Croatia, August-September 2006.
[14] J.-C. Tuan, T.-S. Chang, and C.-W. Jen, “On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 1, pp. 61–72, 2002.
[15] T. Dias, N. Roma, and L. Sousa, “Low power distance measurement unit for real-time hardware motion estimators,” in Proceedings of International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS ’06), pp. 247–255, Montpellier, France, September 2006.
[16] L. Sousa and N. Roma, “Low-power array architectures for motion estimation,” in Proceedings of IEEE International Workshop on Multimedia Signal Processing (MMSP ’99), pp. 679–684, Copenhagen, Denmark, September 1999.
[17] Xilinx Inc., “User Guide. v1.1.1,” ML310 User Guide for Virtex-II Pro Embedded Development Platform, October 2004.
[18] Virtual Silicon Technology Inc., “eSi-Route/11™ high performance standard cell library (UMC 0.18μm),” Tech. Rep. v2.4., November 2001.
[19] A. Berić, R. Sethuraman, H. Peters, J. van Meerbergen, G. de Haan, and C. A. Pinto, “A 27 mW 1.1 mm² motion estimator for picture-rate up-converter,” in Proceedings of the 17th International Conference on VLSI Design (VLSI ’04), vol. 17, pp. 1083–1088, Mumbai, India, January 2004.
Research Article

A Systematic Approach to Design Low-Power Video Codec Cores

Kristof Denolf,1 Adrian Chirila-Rus,2 Paul Schumacher,2 Robert Turney,2 Kees Vissers,2 Diederik Verkest,1,3,4 and Henk Corporaal5

1 D6, IMEC, Kapeldreef 75, 3001 Leuven, Belgium
2 Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124-3400, USA
3 Department of Electrical Engineering, Katholieke Universiteit Leuven (KUL), 3001 Leuwen, Belgium
4 Department of Electrical Engineering, Vrije Universiteit Brussel (VUB), 1050 Brussel, Belgium
5 Faculty of Electrical Engineering, Technical University Eindhoven, Den Dolech 2, 5612 AZ Eindhoven, The Netherlands

Received 2 June 2006; Revised 7 December 2006; Accepted 5 March 2007

The higher resolutions and new functionality of video applications increase their throughput and processing requirements. In contrast, the energy and heat limitations of mobile devices demand low-power video cores. We propose a memory and communication centric design methodology to reach an energy-efficient dedicated implementation. First, memory optimizations are combined with algorithmic tuning. Then, a partitioning exploration introduces parallelism using a cyclo-static dataflow model that also expresses implementation-specific aspects of communication channels. Towards hardware, these channels are implemented as a restricted set of communication primitives. They enable an automated RTL development strategy for rigorous functional verification. The FPGA/ASIC design of an MPEG-4 Simple Profile video codec demonstrates the methodology. The video pipeline exploits the inherent functional parallelism of the codec and contains a tailored memory hierarchy with burst accesses to external memory. 4CIF encoding at 30 fps, consumes 71 mW in a 180 nm, 1.62 V UMC technology.

Copyright © 2007 Kristof Denolf et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

New video appliances, like cellular videophones and digital cameras, not only offer higher resolutions, but they also support the latest coding/decoding techniques utilizing advanced video tools to improve the compression performance. These two trends continuously increase the algorithmic complexity and the throughput requirements of video coding applications and complicate the challenges to reach a real-time implementation. Moreover, the limited battery power and heat dissipation restrictions of portable devices create the demand for a low-power design of multimedia applications. Their energy efficiency needs to be evaluated from the system including the off-chip memory, as its bandwidth and size has a major impact on the total power consumption and the final throughput.

In this paper, we propose a dataflow oriented design approach for low-power block based video processing and apply it to the design of a MPEG-4 part 2 Simple Profile video encoder. The complete flow has a memory focus motivated by the data dominated nature of video processing, that is, the data transfer and storage has a major impact on the energy efficiency and on the achieved throughput of an implementation [1–3]. We concentrate on establishing the overall design flow and show how previously published design steps and concepts can be combined with the parallelization and verification support. Additionally, the barrier to the high energy efficiency of dedicated hardware is lowered by an automated RTL development and verification environment reducing the design time.

The energy efficiency of a real-time implementation depends on the energy spent for a task and the time budget required for this task. The energy delay product [4] expresses both aspects. The nature of the low-power techniques and their impact on the energy delay product evolve while the designer goes through the proposed design flow. The first steps of the design flow are generic (i.e., applicable to other types of applications than block-based video processing). They combine memory optimizations and algorithmic tuning at the high-level (C code) which improve the data
locality and reduce the computations. These optimizations improve both factors of the energy delay product and prepare the partitioning of the system. Parallelization is a well-known technique in low-power implementations: it reduces the delay per task while keeping the energy per task constant. The partitioning exploration step of the design flow uses a Cyclo-Static DataFlow (CSDF, [5]) model to support the buffer capacities sizing of the communication channels between the parallel tasks. The queues implementing these communication channels restrict the scope of the design flow to block based processing as they mainly support transferring blocks of data. The lowest design steps focus on the development of dedicated hardware accelerators as they enable the best energy-efficiency [6, 7] at the cost of flexibility. Since specialized hardware reduces the overhead work a more general processor needs to do, both energy and performance can be improved [4]. For the MPEG-4 Simple Profile video encoder design, applying the proposed strategy results in a fully dedicated video pipeline consuming only 71 mW in a 180 nm, 1.62 V technology when encoding 4CIF at 30 fps.

This paper is organized as follows. After an overview of related work, Section 3 introduces the methodology. The remaining sections explain the design steps in depth and how to apply them on the design of a MPEG-4 Simple Profile encoder. Section 4 first introduces the video encoding algorithm, and then sets the design specifications and summarizes the high-level optimizations. The resulting localized system is partitioned in Section 5 by first describing it as a CSDF model. The interprocess communication is realized by a limited set of communication primitives. Section 6 develops for each process a dedicated hardware accelerator using the RT-level development and verification strategy to reduce the design time. The power efficiency of the resulting video encoder core is compared to state of the art in Section 7. The conclusions are the last section of the paper.

2. RELATED WORK

The design experiences of [8] on image/video processing indicate the required elements in rigorous design methods for the cost efficient hardware implementation of complex embedded systems: higher abstraction levels and extended functional verification. An extensive overview of specification, validation, and synthesis approaches to deal with these aspects is given in [9]. The techniques for power aware system design [10] are grouped according to their impact on the energy delay product in [4]. Our proposed design flow assigns them to a design step and identifies the appropriate models. It combines and extends known approaches and techniques to obtain a low-power implementation.

The Data Transfer and Storage Exploration (DTSE) [11, 12] presents a set of loop and dataflow transformations, and memory organization tasks to improve the data locality of an application. In this way, the dominating memory cost factor of multimedia processing is tackled at the high level. Previously, we combined this DTSE methodology with algorithmic optimizations complying with the DTSE rules [13]. This paper also makes extensions at the lower levels with a partitioning exploration matched towards RTL development. Overall, we now have a complete design flow dealing with the dominant memory cost of video processing focused on the development of dedicated cores.

Synchronous Dataflow (SDF, [14]) and Cyclo-Static Dataflow (CSDF, [5]) models of computation match well with the dataflow dominated behavior of video processing. They are good abstraction means to reason on the parallelism required in a high-throughput implementation. Other works make extensions to (C)SDF to describe image [15] and video [16, 17] applications. In contrast, we use a specific interpretation that preserves all analysis potential of the model. Papers describing RTL code generation from SDF graphs use either a centralized controller [18–20] or a distributed control system [21, 22]. Our work belongs to the second category, but extends the FIFO channels with other communication primitives that support our extensions to CSDF and also retain the effect of the high-level optimizations.

The selected and clearly defined set of communication primitives is the key element of the proposed design flow. It allows to exploit the principle of separation of communication and computation [23] and enables an automated RT-level development and verification strategy that combines simulation with fast prototyping. The Mathworks Simulink/Xilinx SystemGenerator has a similar goal at the level of datapaths [24]. Their basic communication scheme can benefit from the proposed communication primitives to raise the abstraction level. Other design frameworks offer simulation and FPGA emulation [25], with improved signal visibility in [26], at different abstraction levels (e.g., transaction level, cycle true and RTL simulation) that trade accuracy for simulation time. Still, the RTL simulation speed is insufficient to support exhaustive testing and the behavior of the final system is not repeated at higher abstraction levels. Moreover, there is no methodological approach for RTL development and debug. Amer et al. [27] describes upfront verification using SystemC and fast prototyping [28] on an FPGA board, but the coupling between both environments is not explained.

The comparison of the hardware implementation results of building a MPEG-4 part 2 Simple Profile video encoder according to the proposed design flow is described in Section 7.

3. DESIGN FLOW

The increasing complexity of modern multimedia codecs or wireless communications makes a direct translation from C to RT-level impossible: it is too error-prone and it lacks a modular verification environment. In contrast, refining the system through different abstraction levels covered by a design flow helps to focus on the problems related to each design step and to evolve gradually towards a final, energy efficient implementation. Additionally, such design approach shortens the design time: it favors design reuse and allows structured verification and fast prototyping.

The proposed design flow (Figure 1) uses different models of computation (MoC), adapted to the particular design step, to help the designer reasoning about the properties of the system (like memory hierarchy, parallelism, etc.) while a
programming model (PM) provides the means to describe it. The flow starts from a system specification (typically provided by an algorithm group or standardization body like MPEG) and gradually refines it into the final implementation: a netlist with an associated set of executables. Two major phases are present: (i) a sequential phase aiming to reduce the complexity with a memory focus and (ii) a parallel phase in which the application is divided into parallel processes and mapped to a processor or translated to RTL.

The previously described optimizations [11, 13] of the sequential phase transform the application into a system with localized data communication and processing to address the dominant data cost factor of multimedia. This localized behavior is the link to the parallel phase: it allows to extract a cyclo-static dataflow model to support the partitioning (see Section 5.1) and it favors small data units perfectly fitting the block FIFO of the limited but sufficient set of Communication Primitives (CPs, Section 5.2) supporting interprocess data transfers. At the lower level, these CPs can be realized as zero-copy communication channels to limit their energy consumption.

The gradual refinement of the system specification as executable behavioral models, described in a well-defined PM, yields a reference used throughout the design that, combined with a testbench, enables profound verification in all steps. Additionally, exploiting the principle of separation of communication and computation in the parallel phase, allows a structured verification through a combination of simulation and fast prototyping (Section 6).

### 3.1. Sequential phase

The optimizations applied in this first design phase are performed on a sequential program (often C code) at the higher design-level offering the best opportunity for the largest complexity reductions [4, 10, 29]. They have a positive effect on both terms of the energy delay product and are to a certain degree independent of the final target platform [11, 13]. The ATOMIUM tool framework [30] is used intensively in this phase to validate and guide the decisions.

**Preprocessing and analysis (see Section 4)**

The preprocessing step restricts the reference code to the required functionality given a particular application profile and prepares it for a meaningful first complexity analysis that identifies bottlenecks and initial candidates for optimization. Its outcome is a golden specification. During this first step, the testbench triggering all required video tools, resolutions, frame rates, and so forth is fixed. It is used throughout the design for functional verification and it is automated by scripts.

**High-level optimizations (see Section 4)**

This design step combines algorithmic tuning with dataflow transformations at the high-level to produce a memory-optimized specification. Both optimizations aim at (1) reducing the required amount of processing, (2) introducing data locality, (3) minimizing the data transfers (especially to large memories), and (4) limiting the memory footprint. To also enable data reuse, an appropriate memory hierarchy is selected. Additionally, the manual rewriting performed in this step simplifies and cleans the code.

### 3.2. Parallel phase

The second phase selects a suited partitioning and translates each resulting process to HDL or optimizes it for a chosen processor. Introducing parallelism keeps the energy per operation constant while reducing the delay per operation. Since the energy per operation is lower for decreased performance (resulting from voltage-frequency scaling), the parallel solution will dissipate less power than the original solution [4]. Dedicated hardware can improve both energy and performance. Traditional development tools are completed with the automated RTL environment of Section 6.

**Partitioning (see Section 5)**

The partitioning derives a suited split of the application in parallel processes that, together with the memory hierarchy, defines the system architecture. The C model is reorganized to closely reflect this selected structure. The buffer sizes of the interprocess communication channels are calculated based on the relaxed cyclo-static dataflow [5] (Section 5.1) MoC. The PM is mainly based on a message passing system and is defined as a limited set of communication primitives (Section 5.2).

**RTL development and software tuning (see Section 6)**

The RTL describes the functionality of all tasks in HDL and tests each module including its communication separately to verify the correct behavior (Section 6). The software (SW) tuning adapts the remaining code for the chosen processor(s)
through processor specific optimizations. The MoC for the RTL is typically a synchronous or timed one. The PM is the same as during the partitioning step but is expressed using an HDL language.

**Integration (see Section 6)**

The integration phase first combines multiple functional blocks gradually until the complete system is simulated and mapped on the target platform.

### 4. PREPROCESSING AND HIGH-LEVEL OPTIMIZATIONS

The proposed design flow is further explained while it is applied on the development of a fully dedicated, scalable MPEG-4 part 2 Simple Profile video codec. The encoder and decoder are able to sustain, respectively, up to 4CIF (704 × 576) at 30 fps and XSGA (1280 × 1024) at 30 fps or any multistream combination that does not supersede these throughputs. The similarity of the basic coding scheme of a MPEG-4 part 2 video codec to that of other ISO MPEG and ITU-T standards (even more recent ones) makes it a relevant design is described in [31]. The decoder sequential phase are given in [13]. The decoder design is described in [31].

The MPEG-4 part 2 video codec [32] belongs to the class of lossy hybrid video compression algorithms [33]. The architecture of Figure 5 also gives a high-level view of the encoder. A frame is divided in macroblocks, each containing 6 blocks of 8 × 8 pixels: 4 luminance and 2 chrominance blocks. The Motion Estimation (ME) exploits the temporal redundancy by searching for the best match for each new input block in the previously reconstructed frame. The motion vectors define this relative position. The remaining error information after Motion Compensation (MC) is decorrelated spatially using a DCT transform and is then Quantized (Q). The inverse operations Q⁻¹ and IDCT (completing the texture coding chain) and the motion compensation reconstruct the frame as generated at the decoder side. Finally, the motion vectors and quantized DCT coefficients are variable length encoded. Completed with video header information, they are structured in packets in the output buffer. A rate control algorithm sets the quantization degree to achieve a specified average bitrate and to avoid over or under flow of this buffer. The testbench described Table 1 is used at the different design stages. The 6 selected video samples have different sizes, framrates and movement complexities. They are compressed at various bitrates. In total, 20 test sequences are defined in this testbench.

| Test sequence     | Frame rates | Frame sizes | Bitrates (bps) |
|-------------------|-------------|-------------|----------------|
| Mother & daughter | 15, 30      | QCIF, CIF   | 75 k, 100 k, 200 k |
| Foreman           | 15, 30      | QCIF, CIF   | 200 k, 400 k, 800 k |
| Calendar & mobile | 15, 30      | QCIF, CIF   | 500 k, 1.5 M, 3 M |
| Harbour           | 15, 30      | QCIF, CIF, 4CIF | 100 k, 500 k, 2 M |
| Crew              | 15, 30      | QCIF, CIF, 4CIF | 200 k, 1 M, 4 M, 6 M |
| City              | 15, 30      | QCIF, CIF, 4CIF | 200 k, 1 M, 4 M, 6 M |

Exploits the freedom available at the encoder side to trade a limited amount of compression performance (less than 0.5 dB, see [13]) for a large complexity reduction. Two types of algorithmic optimization are applied: modifications to enable macroblock based processing and tuning to reduce the required processing for each macroblock. The development of a predictive rate control [35], calculating the mean absolute deviation by only using past information belongs to the first category. The development of directional squared search motion estimation [36] and the intelligent block processing in the texture coding [13] are in the second class.

#### Algorithmic tuning

Exploits the freedom available at the encoder side to trade a limited amount of compression performance (less than 0.5 dB, see [13]) for a large complexity reduction. Two types of algorithmic optimization are applied: modifications to enable macroblock based processing and tuning to reduce the required processing for each macroblock. The development of a predictive rate control [35], calculating the mean absolute deviation by only using past information belongs to the first category. The development of directional squared search motion estimation [36] and the intelligent block processing in the texture coding [13] are in the second class.

#### Memory optimizations

In addition to the algorithmic tuning reducing the ME's number of searched positions, a two-level memory hierarchy (Figure 2) is introduced to limit the number of accesses to the large frame sized memories. As the ME is intrinsically a localized process (i.e., the matching criterion computations repeatedly access the same set of neighboring pixels), the heavily used data is preloaded from the frame-sized memory to smaller local buffers. This solution is more efficient as soon as the cost of the extratransfers is balanced by the advantage of using smaller memories. The luminance information of the previous reconstructed frame required by the motion estimation/compensation is stored in a buffer Y. The search area buffer is a local copy of the values repetitively accessed during the motion estimation. This buffer is circular in the horizontal direction to reduce the amount of writes during the
updated in this buffer. Both chrominance components have a similar buffer-U/V to copy the data of the previously reconstructed frame needed by the motion compensation. In this way, the newly coded macroblocks can be immediately stored in the frame memory and a single reconstructed frame is sufficient to support the encoding process. This reconstructed frame memory has a block-based data organization to enable burst-oriented reads and writes. Additionally, skipped blocks with zero motion vectors do not need to be stored in the single reconstructed frame memory, as its content did not change with respect to the previous frame.

To further increase data locality, the encoding algorithm is organized to support macroblock-based processing. The motion compensation, texture coding, and texture update work on a block granularity. This enables an efficient use of the communication primitives. The size of the blocks in the block FIFO queues is minimized (only blocks or macroblocks), off-chip memory accesses are reduced as the reconstructed frame is maximally read once and written once per pixel and its accesses are grouped in bursts.

5. PARTITIONING EXPLORATION

The memory-optimized video encoder with localized behavior mainly processes data structures (e.g., (macro)blocks, frames) rather than individual data samples as in a typical DSP system. In such a processing environment the use of dataflow graphs is a natural choice. The next subsection briefly introduces Cyclo-Static DataFlow (CSDF) [5], explains its interpretation and shows how buffer sizes are calculated. Then the set of CPs supporting this CSDF model are detailed. Finally, the partitioning process of the encoder is described.

5.1. Partitioning using cyclo-static dataflow techniques

CSDF is an extension of Static DataFlow (SDF [14]). These dataflow MoCs use graphical dataflow to represent the application as a directed graph, consisting of actors (processes) and edges (communication) between them [37]. Each actor produces/consumes tokens according to firing rules, specifying the amount of tokens that need to be available before the actor can execute (fire). This number of tokens can change periodically resulting in a cyclo-static behavior.

The data-driven operation of a CSDF graph allows for an automatic synchronization between the actors: an actor cannot be executed prior to the arrival of its input tokens. When a graph can run without a continuous increase or decrease of tokens on its edges (i.e., with finite buffers) it is said to be consistent and live.

5.1.1. CSDF interpretation

To correctly represent the behavior of the final implementation, the CSDF model has to be build in a specific way. First, the limited size and blocking read and blocking write behavior of the synchronizing communication channels (see Section 5.2), are expressed in CSDF by adding a backward edge representing the available buffer space [37]. In this way, firing an actor consists of 3 steps: (i) acquire: check the availability of the input tokens and output tokens buffer space, (ii) execute the code of the function describing the behavior of the actor (accessing the data in the container of the actor) and (iii) release: close the production of the output tokens and the consumption of the input tokens.

Second, as the main focus of the implementation efficiency is on the memory cost, the restrictions on the edges are relaxed: partial releases are added to the typically random accessible data in the container of a token. These partial releases enable releasing only a part of the acquired tokens to support data re-use. A detailed description of all relaxed edges is outside the scope of this paper. Section 5.2 realizes the edges as two groups: synchronizing CPs implementing the normal CSDF edges and nonsynchronizing CPs for the relaxed ones.

Finally, the monotonic behavior of a CSDF graph [38] allows to couple the temporal behavior of the model to the final implementation. This monotonic execution assures that smaller Response Times (RTs) of actors can only lead to an equal or earlier arrival of tokens. Consequently, if the buffer size calculation of the next section is based on worst-case RTs and if the implemented actor never exceeds this worst-case RT, then throughput of the implementation is guaranteed.

5.1.2. Buffer size calculation

Reference [5] shows that a CSDF-graph is fully analyzable at design time: after calculating the repetition vector \( q \) for the
This allows to exploit the principle of separation of communication building blocks for the program- ming model that is available at the di- rectional model can be isolated and translated to lower levels, wherein the targeted fully dedicated imple- mentation, the desired schedule operates in a fully parallel and pipelined way. It is as- sumed that every actor runs on its own processor (i.e., no time multiplexing and sufficient resources) to maximize the RT of each actor. This inherently eases the job of the designer handwriting the RTL during the next design step and yields better synthesis results. Consequently, the RT of each actor \( A \) is inversely proportional to its repetition rate \( q_A \) and can be expressed relatively to the RT of an actor \( S \),

\[
RT_A = \frac{RT_S q_S}{q_A}. \tag{1}
\]

Under these assumptions and with the CSDF interpretation presented above, the buffer size equals the maximum amount of the acquired tokens while executing the desired schedule. Once this buffer sizing is completed, the system has a self-timed behavior.

### 5.2. Communication primitives

The communication primitives support the inter- actor/process(or) communication and synchronization methods expressed by the edges in the CSDF model. They form a library of communication building blocks for the programming model that is available at the different abstraction levels of the design process. Only a limited set of strictly defined CPs are sufficient to support a video codec implementation. This allows to exploit the principle of separation of communication and computation [23] in two ways: first to create and test the CPs separately and second to cut out a functional module at the borders of its I/O (i.e., the functional component and its CPs) and develop and verify it individually (see Section 6). In this way, functionality in the high-level functional model can be isolated and translated to lower levels, while the component is completely characterized by the input stimuli and expected output.

All communication primitives are memory elements that can hold data containers of the tokens. Practically, depending on the CP size, registers or embedded RAM implement this storage. Two main groups of CPs are distinguished: synchronizing and nonsynchronizing CPs. Only the former group provides synchronization support through its blocking read and blocking write behavior. Consequently, the proposed design approach requires that each process of the system has at least one input synchronizing CP and at least one output synchronizing CP. The minimal compliance with this condition allows the system to have a self-timed execution that is controlled by the depth of the synchronizing CPs, sized according to the desired schedule in the partitioning step (Section 5.1.2).

#### 5.2.1. Synchronizing/token-based communication primitives

The synchronizing CPs signal the presence of a token next to the storage of the data in the container to support implementing the blocking read and blocking write of the CSDF MoC (Section 5.1). Two types are available: a scalar FIFO and a block FIFO (Figure 3).

The most general type, the block FIFO represented in Figure 3, passes data units (typically a (macro)block) between processes. It is implemented as a first in first out queue of data containers. The data in the active container within the block FIFO can be accessed randomly. The active container is the block that is currently produced/consumed on the production/consumption side. The random access capability of the active container requires a control signal (\(\text{op}_\text{mode} \)) to follow the operations: (1) NOP, (2) read, (3) write, and (4) commit. The commit command indicates the releasing of the active block (in correspondence to last steps of the actor firing in the CSDF model of Section 5.1.1).

The block FIFO offers interesting extrafeatures.

(i) Random access in container allowing to produce values in a different order than they are consumed, like the (zigzag) scan order for the (I)DCT.

(ii) The active container can be used as scratch pad for local temporary data.

(iii) Transfer of variable size data as not all data needs to be written.

The scalar FIFO is a simplified case of the block FIFO, where a block contains only a single data element and the control signal is reduced to either read or write.

#### 5.2.2. Nonsynchronizing communication primitives

The main problem introduced by the token based processing is the impossibility of reusing data between two processes and the incapability to efficiently handle parameters that are not aligned on data unit boundaries (e.g., Frame/Slice based parameters). In order to enable a system to handle these exceptional cases expressed by relaxed edges in the CSDF model (Section 5.1.1), the following communication primitives are introduced: shared memory and configuration registers. As they do not offer token support, they can only be used between processes that are already connected (indirectly) through synchronizing CPs.
Table 2: Detailed information of the actors in the encoder CSDF graph.

| Actor name                | Functionality                                   | Repetition rate | WCRT (μs) |
|---------------------------|-------------------------------------------------|-----------------|-----------|
| Input control             | Load the new video inputs                       | 1               | 21.0      |
| Copy control              | Fill the memory hierarchy                       | 1               | 21.0      |
| Motion estimation         | Find the motion vectors                         | 1               | 21.0      |
| Motion compensation       | Get predicted block and calculate error          | 6               | 3.5       |
| Texture coding            | Transform, quantization and inverse             | 6               | 3.5       |
| Texture update            | Add and clip compensated and predicted blocks   | 6               | 3.5       |
| Entropy coding            | AC/DC, MV prediction and VLC coding            | 1               | 21.0      |
| Bitstream packetization   | Add headers and compose the bitstream           | ≤ 1 ≥ 21.0      |           |

Figure 4: Nonsynchronizing communication primitives.

**Shared memory**

The shared memory, presented in Figure 4(a), is used to share pieces of a data array between two or more processes. It typically holds data that is reused potentially multiple times (e.g., the search area of a motion estimation engine). Shared memories are conceptually implemented as multi-port memories, with the number of ports depending on the amount of processing units that are simultaneously accessing it.

Larger shared memories, with as special case external memory, are typically implemented with a single port. A memory controller containing an arbiter handles the accesses from multiple processing units.

**Configuration registers**

The configuration registers (Figure 4(b)) are used for unsynchronized communication between functional components or between hardware and remaining parts in the software. They typically hold the scalars configuring the application or the parameters that have a slow variation (e.g., frame parameters). The configuration registers are implemented as shadow registers.

**5.3. Video pipeline architecture**

The construction of an architecture suited for the video encoder starts with building a CSDF graph of the high-level optimized version. The granularity of the actors is chosen fine enough to enable their efficient implementation as hardware accelerator. Eight actors (see Figure 5) are defined for the MPEG-4 encoder. Table 2 contains a brief description of the functionality of each actor and its repetition rate. Adding the edges to the dataflow graph examines the communication between them and the required type of CP. The localized processing of the encoder results in the use of block FIFOs exchanging (macro)block size data at high transfer rates and to synchronize all actors. The introduced memory hierarchy requires shared memory CPs. At this point of the partitioning, all CPs of Figure 5 correspond to an edge and have an unlimited depth.

By adding a pipelined and parallel operation as desired schedule, the worst-case response time (WCRT) of each actor is obtained with (1) for a throughput of 4CIF at 30 fps (or 47520 macroblocks per second) and listed in Table 2. These response times are used in the lifetime analysis (of Section 5.1.2) to calculate the required buffer size of all CPs.

The resulting video pipeline has a self-timed behavior. The concurrency of its processes is assured by correctly sizing these communication primitives. In this way, the complete pipeline behaves like a monolithic hardware accelerator. To avoid interface overheads [39], the software orchestrator calculates the configuration settings (parameters) for all functional modules on a frame basis. Additionally, the CPs are realized in hardware as power efficient dedicated zero-copy communication channels. This avoids first making a local copy at the producer, then reading it back to send it over a bus or other communication infrastructure and finally storing it in another local buffer at the consumer side.

6. RTL DEVELOPMENT AND VERIFICATION ENVIRONMENT

The proposed RTL development and verification methodology simplifies the HW description step of the design flow. It covers the HDL translation and verification of the individual functional components and their (partial) composition into a system. The separation of communication and computation permits the isolated design of a single functional module. Inserted probes in the C model generate the input stimuli and the expected output characterizing the behavior of the block. As the number of stimuli required to completely test a functional module can be significant, the development
environment supports simulation as well as testing on a prototyping or emulation platform (Figure 6). While the high signal visibility of simulation normally produces long simulation times, the prototyping platform supports much faster and more extensive testing with the drawback of less signal observability.

Reinforcing the communication primitives on the software model and on the hardware block allows the generation of the input stimuli and of the expected output from the software model, together with a list of ports grouped in the specification file (SPEC). The SPEC2VHDL tool generates, based on this specification (SPEC) file, the VHDL testbenches, instantiates the communication primitives required by the block, and also generates the entity and an empty architecture of the designed block. The testbench includes a VHDL simulation library that links the stimuli/expected output files with the communication primitives. In the simulation library basic control is included to trigger full/empty behavior. The communication primitives are instantiated from a design library, which will also be used for synthesis. At this point the designer can focus to manually complete only the architecture of the block.

As the user finishes the design of the block, the extensive testing makes the simulation time a bottleneck. In order to speed up the testing phase, a seamless switch to a fast prototyping platform based on the same SPEC file and stimuli/expected output is supported by SPEC2FPGA. This includes the generation of the software application, link to the files, and low-level platform accesses based on a C/C++ library. Also the platform/FPGA required interfaces are generated together with the automatic inclusion of the previously generated entity and implemented architecture.

To minimize the debug and composition effort of the different functional blocks, the verification process uses the traditional two phases: first blocks are tested separately and then they are gradually combined to make up the complete system. Both phases use the two environments of Figure 6.

The combination of the two above described tools, creates a powerful design and verification environment. The designer can first debug and correct errors by using the high signals visibility of the simulation tools. To extensively test the developed functional module, he uses the speed of the prototyping platform to identify an error in a potential huge test bed. As both simulation and hardware verification setups are functionally identical, the error can be identified on the prototyping platform with a precision that will allow a reasonable simulation time (e.g., sequence X, frame Y) in view of the error correction.

7. IMPLEMENTATION RESULTS

Each actor of Figure 5 is individually translated to HDL using the development and verification approach described in the previous section. The partitioning is made in such a way that the actors are small enough to allow the designer to come up with a manual RTL implementation that is both energy and throughput efficient. Setting the target operation frequency to 100 MHz, results in a budget of 2104 cycles per firing for the actors with a repetition rate of 1 and a budget of 350 cycles for the actors with a repetition rate of 6 (see Table 2). The throughput is guaranteed when all actors respect this worst-case execution time. Because of the temporal monotonic behavior (Section 5.1.1) of the self-timed executing pipeline, shorter execution times can only lead to an equal or higher performance.

The resulting MPEG-4 part 2 SP encoder is first mapped on the Xilinx Virtex-II 3000 (XC2V3000-4) FPGA available on the Wildcard-II [40] used as prototyping/demonstration platform during verification. Second, the Synopsys tool suite is combined with Modelsim to evaluate the power efficiency and size of an ASIC implementation.

7.1. Throughput and Size

Table 3 lists the operation frequencies required to sustain the throughput of the different MPEG-4 SP levels. The current design can be clocked up to 100 MHz both on the FPGA\(^1\) and on the ASIC, supporting 30 CIF frames per second, exceeding the level 5 requirements of the MPEG standard [41]. Additionally, the encoder core supports processing of multiple video sequences (e.g., 4 × 30 CIF frames per second). The user can specify the required maximum frame size through the use of HDL generics to scale the design according to his needs (Table 3).

---

\(^{1}\) Frequency achieved for Virtex4 speed grade −10. Implementation on Virtex2 or Spartan3 may not reach this operating frequency.

---

**Table 3: Required operation frequency, off-chip data rates (encoding the City reference video sequence) and FPGA resource consumption for different levels.**

| Throughput (fps) & level | Operation frequency (MHz) | External memory (kB) | 32-bit external transfers (10\(^6\)/s) | FPGA resources |
|--------------------------|---------------------------|----------------------|--------------------------------------|----------------|
|                          | Measured | Not optimized\(^2\) | Measured | Not optimized\(^3\) | Slices | LUTs | FFs | BRAMs | Mults |
| 15 QCIF (L1)             | 3.2      | 4.0                  | 37       | 0.28 | 0.29 | 8303 | 12630 | 6093 | 16  | 17 |
| 15 CIF (L2)              | 12.9     | 17.6                 | 149      | 1.14 | 1.14 | 9019 | 12778 | 6335 | 23  | 17 |
| 30 CIF (L3)              | 25.6     | 35.2                 | 149      | 2.25 | 2.28 | 9019 | 12778 | 6335 | 23  | 17 |
| 30 4CIF (>L5)            | 100.7    | 140.6                | 594      | 9.07 | 9.12 | 9353 | 13260 | 6575 | 36  | 17 |

\(^{2}\) not optimized = proposed ME algorithm (directional squared search) without early stop criteria.

\(^{3}\) not optimized = reading and writing every sample once.
Figure 5: MPEG-4 simple profile encoder block diagram.

Figure 6: Development and verification environment.
Table 4: Hardware characteristics of the encoder core

| Encoder core          | Power distribution (4CIF 30 fps) |
|-----------------------|----------------------------------|
|                       | part | power (mW) | share % |
|                       | block FIFO | 16.4 | 23.1 |
|                       | shared memory | 13.3 | 18.7 |
|                       | scalar FIFO | 3.8 | 5.4 |
|                       | texture coding | 20.7 | 29.2 |
|                       | motion estimation | 7.4 | 10.5 |
|                       | entropy coder | 3.4 | 4.7 |
|                       | other | 6.0 | 8.4 |

Table 5: Characteristics of state-of-the-art MPEG-4 part 2 video implementations.

| Design     | Throughput (Mpixels/s) | Process (nm, V) | Frequency (MHz) | Power (mW) | Area (kGates) | Off-chip SDRAM (Byte) | External accesses per pixel | Scaled Power (mW) | Scaled Through-put (Mpixels/s) | Scaled energy per pixel (nJ/pixel) | Scaled energy delay product (nJ·μs) |
|------------|------------------------|-----------------|----------------|-----------|--------------|---------------------|------------------------|----------------|-------------------------------|------------------------------------|-----------------------------------|
| [44]       | 3.0 (L3)               | 180, 1.5        | 36             | 116       | 170          | 43                  | —                     | 161 k          | —                             | 3.0                                | 12.5                              |
| [45]       | 1.5 (L2)               | 180, 1.5        | 29             | 400       | —            | 2 M                | —                     | 128 k        | —                             | 19.1                              | 12.5                              |
| [47]       | 0.4 (L1)               | 180, 1.5        | 170            | —         | 128          | —                  | —                     | 2 M           | —                             | 447.2                             | 1176.3                            |
| [46]       | 1.5 (L2)               | 130, 1.5        | 160            | 3000      | —            | —                  | —                     | 2 M           | —                             | 279.3                             | 254.3                             |
| [43]       | 3.0 (L3)               | 350, 3.3        | 257            | 207       | 39           | 2 M                | —                     | 14             | 2.7                           | 1.1                                | 1.9                               |
| [42]       | 27.6 (>L5)             | 130, 1.3        | 120            | 390       | 80           | —                  | —                     | 2.6 M         | 17                            | 23.0                              | 13.3                              |
| [51]       | 9.2 (L4)               | 130, 1.2        | 50             | 800       | —            | —                  | —                     | 8.3            | 13.3                          | 31.0                              | 4.1                               |
| [39]       | 9.2 (L4)               | 130, 1.2        | 9.6            | 170       | —            | —                  | —                     | 31             | 10.0                          | 21.9                              | 2.1                               |
| [52]       | 9.2 (L4)               | 180, 1.4        | 18             | 201       | 36           | —                  | —                     | 9.9            | 2.1                           | 21.9                              | 0.2                               |
| This work  | 12.2 (>L5)             | 180, 1.6        | 71             | 87        | 400          | 594 k              | 2                     | 61             | 11.3                          | 5.4                               | 0.5                               |

4 Moved on-chip using embedded DRAM.
5 Assuming 1 gate = 4 transistors.

7.2. Memory requirements

On-chip BRAM (FPGA) or SRAM (ASIC) is used to implement the memory hierarchy and the required amount scales with the maximum frame size (Table 3). Both the copy controller (filling the bufferYUV and search area, see Figure 5) and the texture update make 32 bit burst accesses (of 64 bytes) to the external memory, holding the reconstructed frame with a block-based data organization. At 30 4CIF frames per second, this corresponds in worst-case to 9.2 Mtransfers per second (as skipped blocks are not written to the reconstructed frame, the values of the measured external transfers in Table 3 are lower). In this way, our implementation minimizes the off-chip bandwidth with at least a factor of 2.5 compared to [42–45] without embedding a complete frame memory as done in [46, 47] (see also Table 5). Additionally, our encoder only requires the storage of one frame in external memory.

7.3. Power consumption

Power simulations are used to assess the power efficiency of the proposed implementation. They consist of 3 steps. (1) Synopsys [48] DC Compiler generates a gate level netlist and the list of signals to be monitored for power (forward switching activity file), (2) ModelSim [49] RTL simulation tracks the actual toggles of the monitored signals and produces the back-annotated switching activity file. (3) Synopsys Power Compiler calculates power numbers based on the gate level netlist from step 1 and back annotated switching activity file from step 2. Such prelayout gate-level simulations do not include accurate wire-loads. Internal experiments indicate their impact limited to a 20% error margin. Additionally, I/O power is not included.

Table 4 gives the characteristics of the ASIC encoder core when synthesized for 100 MHz, 4CIF resolution. It also lists the power consumptions while processing the City reference video sequence at different levels when clocked at the corresponding operation frequency of Table 3. These numbers do not include the power of the software orchestrator.

Carefully realizing the communication primitives on the ASIC allows balancing their power consumption compared to the logic (Table 4): banking is applied to the large on-chip bufferYUV and the chip-enable signal of the communication primitives is precisely controlled to shut down the CP ports when idle. Finally, clock-gating is applied to the complete encoder to further reduce the power consumption. To compare the energy efficiency to the available state of the art solutions, the power consumption of all implementations (listed in Table 5) is scaled to the 180 nm, 1.5 V technology node,
Figure 7: Activity diagram of the video pipeline in regime mode. The texture coding (TC) is the critical path in the I frame, motion estimation (ME) is the critical path in the P frame.

using (2), where $P$ is the power, $V_{dd}$ is the supply voltage and $\lambda$ is the feature size. The ITRS roadmap [4, 50] indicates that beyond 130 nm, the quadratic power scaling ($\alpha = \beta = 2$ in (2)) breaks down and follows a slower trend,

$$P_2 = P_1 \left( \frac{V_{dd,2}}{V_{dd,1}} \right)^{\alpha} \left( \frac{\lambda_2}{\lambda_1} \right)^{\beta}.$$  \hfill (2)

Similarly, the throughput $T$ needs to be scaled as it directly related to the achievable operation frequency that depends on the technology node. A linear impact by both the feature size and the supply voltage is assumed in (3),

$$T_2 = T_1 \left( \frac{V_{dd,2}}{V_{dd,1}} \right) \left( \frac{\lambda_1}{\lambda_2} \right).$$ \hfill (3)

The scaled energy per pixel in Table 5 compares the available state of the art MPEG-4 video encoders in a the 180 nm, 1.5 V technology node. The proposed core is clearly more energy efficient than [42, 44, 45, 51]. The power consumption of [43], including a SW controller, is slightly better (note that this work does not include the SW orchestrator). However, taking the off-chip memory accesses into account when evaluating the total system power consumption, the proposed solutions has a (at least) 2.5 times reduced off chip transfer rate, leading to the lowest total power consumption. Even when compared to the ASICs containing embedded DRAM [46, 47], our implementation is more power-efficient as only 1.5 mW is required at L2 (15 CIF fps) to read and write every pixel from the external memory (assuming 1.3 nJ per 32 bit transfer). For L1 and L2 throughput respectively, [46, 47] consume more than 150 mW (Table 5). Our implementation delivers 15 CIF fps (L2) consuming only $9.7 + 1.5 = 11.4$ mW. Using complementary techniques, like a low-power CMOS technology, [39, 52] achieve an even better core energy efficiency. Insufficient details prevent a complete comparison including the external memory transfer cost.

The last column of Table 5 presents the scaled energy delay product. This measure includes the scaled throughput as delay per pixel. The previous observations also hold using this energy delay product, except for the 30 CIF fps of [43] having now a worse result than the proposed encoder. Note that a complete comparison should also include the coding efficiency of the different solutions as algorithmic optimizations (like different ME algorithms) sacrifice compression performance to reduce complexity. Unfortunately, not all referred papers contain the required rate-distortion information to make such evaluation.

7.4. Effect of the high-level optimizations

The algorithmic optimizations of Section 4 reduce the average number of cycles to process a macroblock compared to the worst-case (typically around 25%, see Table 3) and hence lower the power consumption. Figure 7 shows the activity diagram of the pipelined encoder in regime mode. The signal names are the acronyms of the different functional modules in Figure 5. During the I frame (left side of the dotted line in Figure 7), the texture coding (“TC”) is the critical path as it always processes the complete error block. During a P frame (right side of the dotted line in Figure 7), the search for a good match in the previous frame is the critical path (“ME”). The early stop criteria sometimes shorten the amount of cycles required during the motion estimation. When this occurs, often a good match is found, allowing the texture coding to apply its intelligent block processing that reduces the amount of cycles spend to process the macroblock. In this way both critical paths are balanced (i.e., without algorithmic tuning of the texture coding, this process would become the new critical path in a P frame). Additionally, a good match leads to a higher amount of skipped blocks with possible zero motion vectors. Consequently, the algorithmic tuning reduces the amount of processing and data communication, leading to improved power efficiency.
8. CONCLUSIONS

Modern multimedia applications seek to improve the user experience by invoking advanced techniques and by increasing resolutions. To meet the power and heat dissipation limitations of portable devices, their implementation requires a set of various power optimization techniques at different design levels: (i) redefine the problem to reduce complexity, (ii) introduce parallelism to reduce energy per operation and (iii) add specialized hardware as it achieves the best energy efficiency. This philosophy is reflected in the proposed design flow bridging the gap between a high-level specification (typically C) into the final implementation at RTL level. The typical data dominance of multimedia systems motivates the memory and communication focus of the design flow.

In the first phase of the design flow, the reference code is transformed through memory and algorithmic optimizations into a block-based video application with localized processing and data flow. Memory footprint and frame memory accesses are minimized. These optimizations prepare the system for introducing parallelism and provide a reference for the RTL development.

The second design phase starts with partitioning the application in a set of concurrent processes to achieve the required throughput and to improve the energy efficiency. Based on a CSDF model of computation, used in a specific way to also express implementation specific aspects, the required buffers sizes of the graph edges are calculated to obtain a fully pipelined and parallel self-timed execution. By exploiting the principle of separation of communication and computation, each actor is translated individually to HDL while correctly modeling its communication. An automated environment supports the functional verification of such component by combining simulation of the RTL model and testing the RTL implementation on a prototyping platform. The elaborate verification of each single component reduces the debug cycle for integration.

The design methodology is demonstrated on the development of a MPEG-4 Simple Profile video encoder capable of processing 30 4CIF (704 × 576) frames per second or multiple lower resolution sequences. Starting from the reference code, a dedicated video pipeline is realized on an FPGA and ASIC. The core achieves a high degree of concurrency, uses a dedicated memory hierarchy and exploits burst oriented external memory I/O leading to the theoretical minimum of off-chip accesses. The video codec design is scalable with a number of added compile-time parameters (e.g., maximum frame size and number of bitstreams) which can be set by the user to best suit his application. The presented encoder core uses 71 mW (180 nm, 1.62 V UMC technology) when processing 4CIF at 30 fps.

REFERENCES

[1] M. A. Viredaz and D. A. Wallach, “Power evaluation of a handheld computer,” IEEE Micro, vol. 23, no. 1, pp. 66–74, 2003.
[2] A. Lambrechts, P. Raghavan, A. Leroy, et al., “Power breakdown analysis for a heterogeneous NoC platform running a video application,” in Proceedings of the 16th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP’05), pp. 179–184, Samos, Greece, July 2005.
[3] T. Fujisaki, H. Tsuruoka, S. Nomura, et al., “A 63-mW H.264/MPEG-4 audio/visual codec LSI with module-wise dynamic voltage/frequency scaling,” IEEE Journal of Solid-State Circuits, vol. 41, no. 1, pp. 54–62, 2006.
[4] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, “Scaling, power, and the future of CMOS,” in Proceedings of IEEE International Electron Devices Meeting (IEDM’05), p. 7, Washington, DC, USA, December 2005.
[5] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, “Cyclo-static dataflow,” IEEE Transactions on Signal Processing, vol. 44, no. 2, pp. 397–408, 1996.
[6] P. Pirsch, M. Berekovic, H.-J. Stolberg, and J. Jachalsky, “VLSI architectures for MPEG-4,” in Proceedings of International Symposium on VLSI Technology, Systems, and Applications (VTSA’03), pp. 208A–208E, Hsinchu, Taiwan, October 2003.
[7] S.-Y. Chen, Y.-W. Huang, C.-Y. Chen, H. H. Chen, and L.-G. Chen, “Hardware architecture design of video compression for multimedia communication systems,” IEEE Communications Magazine, vol. 43, no. 8, pp. 123–131, 2005.
[8] C.-J. Lian, Y.-W. Huang, H.-C. Fang, Y.-C. Chang, and L.-G. Chen, “JPEG, MPEG-4, and H.264 codec IP development,” in Proceedings of Design, Automation and Test in Europe (DATE’05), vol. 2, pp. 1118–1119, Munich, Germany, March 2005.
[9] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, “Design of embedded systems: formal models, validation, and synthesis,” Proceedings of the IEEE, vol. 85, no. 3, pp. 366–390, 1997.
[10] L. Mazzoni, “Power aware design for embedded systems,” IEE Electronics Systems and Software, vol. 1, no. 5, pp. 12–17, 2003.
[11] E. Cathoor, S. Wuytack, E. de Greef, F. Balasa, L. Nachtegaele, and A. Vandecappelle, Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design, Kluwer Academic Publishers, Norwell, Mass, USA, 1998.
[12] P. R. Panda, E. Cathoor, N. D. Dutt, et al., “Data and memory optimization techniques for embedded systems,” ACM Transactions on Design Automation of Electronic Systems, vol. 6, no. 2, pp. 149–206, 2001.
[13] K. Denolf, C. de Vleeschouwer, R. Turney, G. Lafruit, and J. Bormans, “Memory centric design of an MPEG-4 video encoder,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 5, pp. 609–619, 2005.
[14] E. A. Lee and T. M. Parks, “Dataflow process networks,” Proceedings of the IEEE, vol. 83, no. 5, pp. 773–801, 1995.
[15] A. Davare, Q. Zhu, J. Moondanos, and A. Sangiovanni-Vincentelli, “JPEG encoding on the Intel MXP5800: a platform-based design case study,” in Proceedings of the 3rd IEEE Workshop on Embedded Systems for Real-Time Multimedia (ESTimedia’05), pp. 89–94, New York, NY, USA, September 2005.
[16] H. Hwang, T. Oh, H. Jung, and S. Ha, “Conversion of reference C code to dataflow model: H.264 encoder case study,” in Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC’06), pp. 152–157, Yokohama, Japan, January 2006.
[17] H. Haim, M. Sen, D.-J. Ko, S. S. Bhattacharyya, and W. Wolf, “Mapping multimedia applications onto configurable hardware with parameterized cyclo-static dataflow graphs,” in Proceedings of IEEE International Conference on Acoustics, Speech,
and Signal Processing (ICASSP ’06), vol. 3, pp. 1052–1055, Toulouse, France, May 2006.

[18] M. C. Williamson and E. A. Lee, “Synthesis of parallel hardware implementations from synchronous dataflow graph specifications,” in Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers, vol. 2, pp. 1340–1343, Pacific Grove, Calif, USA, November 1996.

[19] J. Horstmannhoff and H. Meyr, “Efficient building block based RTL code generation from synchronous data flow graphs,” in Proceedings of the 37th Conference on Design Automation (DAC ’00), pp. 552–555, Los Angeles, Calif, USA, June 2000.

[20] H. Jung, K. Lee, and S. Ha, “Efficient hardware controller synthesis for synchronous dataflow graph in system level design,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 4, pp. 423–428, 2002.

[21] J. Dalcolmo, R. Lauwereins, and M. Ade, “Code generation of data dominated DSP applications for FPGA targets,” in Proceedings of the 9th IEEE International Workshop on Rapid System Prototyping, pp. 162–167, Leuven, Belgium, June 1998.

[22] R. Grou-Szabo, H. Ghatts, Y. Savaria, and G. Nicolescu, “Component-based methodology for hardware design of a dataflow processing network,” in Proceedings of the 5th International Workshop on System-on-Chip for Real-Time Applications (IWSoC ’05), pp. 289–294, Banff, Alberta, Canada, July 2005.

[23] K. Keutzer, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli, “System-level design: orthogonalization of concerns and platform-based design,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 12, pp. 1523–1543, 2000.

[24] D. Denning, N. Harold, M. Devlin, and J. Irvine, “Using system generator to design a reconfigurable video encryption system,” in Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL ’03), pp. 980–983, Lisbon, Portugal, September 2003.

[25] Y. Nakamura, K. Hosokawa, I. Kuroda, K. Yoshikawa, and T. Yoshimura, “A fast hardware/software co-verification method for system-on-a-chip by using a C/C++ simulator and FPGA emulator with shared register communication,” in Proceedings of the 41st Design Automation Conference (DAC ’04), pp. 299–304, San Diego, Calif, USA, June 2004.

[26] R. Siripokparirom and F. Mayer-Lindenberg, “Hardware-assisted simulation and evaluation of IP cores using FPGA-based rapid prototyping boards,” in Proceedings of the 15th IEEE International Workshop on Rapid Systems Prototyping, pp. 96–102, Geneva, Switzerland, June 2004.

[27] I. Amer, M. Sayed, W. Badawy, and G. Jullien, “On the way to an H.264 HW/SW reference model: a systemC modeling strategy to integrate selected IP-blocks with the H.264 software reference model,” in Proceedings of IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS ’05), pp. 178–181, Athens, Greece, November 2005.

[28] I. Amer, C. A. Rahman, T. Mohamed, M. Sayed, and W. Badawy, “A hardware-accelerated framework with IP-blocks for application in MPEG-4,” in Proceedings of the 5th International Workshop on System-on-Chip for Real-Time Applications (IWSoC ’05), pp. 211–214, Banff, Alberta, Canada, July 2005.

[29] M. J. Irwin, M. T. Kandemir, N. Vijaykrishnan, and A. Sivasubramaniam, “A holistic approach to system level energy optimization,” in Proceedings of the 10th International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation (PATMOS ’00), pp. 88–107, Springer, Göttingen, Germany, September 2000.

[30] http://www.imec.be/design/atomium/.

[31] P. Schumacher, K. Denolf, A. Chirila-Rus, et al., “A scalable, multi-stream MPEG-4 video decoder for conferencing and surveillance applications,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’05), vol. 2, pp. 886–889, Genova, Italy, September 2005.

[32] “Information technology—generic coding of audio-visual objects—part 2: visual,” ISO/IEC 14496-2:2004, June 2004.

[33] V. Bhaskaran and K. Konstantinides, Image and Video Compression Standards, Algorithms and Architectures, Kluwer Academic Publishers, Boston, Mass, USA, 1997.

[34] “Information technology—generic coding of audio-visual objects—part 5: reference software,” ISO/IEC 14496-5:2001, December 2001.

[35] C. de Vleeschouwer, “Model-based rate control implementation for low-power video communications systems,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 12, pp. 1187–1194, 2003.

[36] C. de Vleeschouwer, T. Nilsson, K. Denolf, and J. Bormans, “Algorithmic and architectural co-design of a motion-estimation engine for low-power video devices,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 12, pp. 1093–1105, 2002.

[37] S. Sriram and S. S. Bhattacharyya, Embedded Multiprocessors: Scheduling and Synchronization, Marcel Dekker, New York, NY, USA, 2000.

[38] M. Wiggers, M. Bekooij, P. Jansen, and G. Smit, “Efficient computation of buffer capacities for multi-rate real-time systems with back-pressure,” in Proceedings of the 4th International Conference on Hardware/software Co-design and System Synthesis (CODES+ISSS ’06), pp. 10–15, Seoul, Korea, October 2006.

[39] T. Rintaluoma, O. Silven, and J. Raekallio, “Interface overheads in embedded multimedia software,” in Proceedings of the 6th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS ’06), pp. 5–14, Samos, Greece, July 2006.

[40] http://www.annapmicro.com/products.html.

[41] ISO/IEC JTC1/SC29/WG11, “Information technology—generic coding of audio-visual objects—part 2: visual, amendment 2: new levels for simple profile,” Tech. Rep. N6496, Redmond, USA, 2004.

[42] H. Yamauchi, S. Okada, T. Watanabe, et al., “An 81MHz, 1280 × 720pixels × 30frames/s MPEG-4 video/audio codec processor,” in Proceedings of IEEE International Solid-State Circuits Conference (ISSCC ’05), vol. 1, pp. 130–589, San Francisco, Calif, USA, February 2005.

[43] Y.-C. Chang, W.-M. Chao, and L.-G. Chen, “Platform-based MPEG-4 video encoder SOC design,” in Proceedings of IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS ’04), pp. 251–256, Austin, Tex, USA, October 2004.

[44] Y. Nakayama, T. Yoshitake, H. Komazaki, et al., “A MPEG-4 video encoder SOC design,” in Proceedings of IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS ’04), pp. 251–256, Austin, Tex, USA, October 2004.

[45] M. Wiggers, M. Bekooij, P. Jansen, and G. Smit, “A SystemC module for multi-stream MPEG-4 video decoder for conferencing and surveillance applications,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’05), vol. 2, pp. 886–889, Genova, Italy, September 2005.

[46] H. Arakida, M. Takahashi, Y. Tsuibo, et al., “A 160 mW, 80 nA standby, MPEG-4 audiovisual LSI with 16 Mb embedded
DRAM and a 5 GOPS adaptive post filter,” in Proceedings of IEEE International Solid-State Circuits Conference (ISSCC ’03), vol. 1, pp. 42–476, San Francisco, Calif, USA, February 2003.

[47] T. Yamada, N. Irie, J. Nishimoto, et al., “A 133 MHz 170 mW 10 μA standby application processor for 3G cellular phones,” in Proceedings of IEEE International Solid-State Circuits Conference (ISSCC ’02), vol. 1, pp. 370–474, San Francisco, Calif, USA, February 2002.

[48] http://www.synopsys.com/.
[49] http://www.model.com/.
[50] http://www.itrs.net/Links/2006Update/FinalToPost/02_Design_2006Update.pdf.

[51] Y. Watanabe, T. Yoshitake, K. Morioka, et al., “Low power MPEG-4 ASP codec IP macro for high quality mobile video applications,” in Proceedings of IEEE International Conference on Consumer Electronics (ICCE ’05), pp. 337–338, Las Vegas, Nev, USA, January 2005.

[52] C.-P. Lin, P.-C. Tseng, Y.-T. Chiu, et al., “A 5mW MPEG4 SP encoder with 2D bandwidth-sharing motion estimation for mobile applications,” in Proceedings of IEEE International Solid-State Circuits Conference (ISSCC ’06), vol. 1, pp. 1626–1635, San Francisco, Calif, USA, February 2006.
Thermal-Aware Scheduling for Future Chip Multiprocessors

Kyriakos Stavrou and Pedro Trancoso
Department of Computer Science, University of Cyprus, 75 Kallipoleos Street, P.O. Box 20537, 1678 Nicosia, Cyprus
Received 10 July 2006; Revised 12 December 2006; Accepted 29 January 2007

The increased complexity and operating frequency in current single chip microprocessors is resulting in a decrease in the performance improvements. Consequently, major manufacturers offer chip multiprocessor (CMP) architectures in order to keep up with the expected performance gains. This architecture is successfully being introduced in many markets including that of the embedded systems. Nevertheless, the integration of several cores onto the same chip may lead to increased heat dissipation and consequently additional costs for cooling, higher power consumption, decrease of the reliability, and thermal-induced performance loss, among others. In this paper, we analyze the evolution of the thermal issues for the future chip multiprocessor architectures and show that as the number of on-chip cores increases, the thermal-induced problems will worsen. In addition, we present several scenarios that result in excessive thermal stress to the CMP chip or significant performance loss. In order to minimize or even eliminate these problems, we propose thermal-aware scheduler (TAS) algorithms. When assigning processes to cores, TAS takes their temperature and cooling ability into account in order to avoid thermal stress and at the same time improve the performance. Experimental results have shown that a TAS algorithm that considers also the temperatures of neighboring cores is able to significantly reduce the temperature-induced performance loss while at the same time, decrease the chip’s temperature across many different operation and configuration scenarios.

Copyright © 2007 K. Stavrou and P. Trancoso. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

The doubling of microprocessor performance every 18 months has been the result of two factors: more transistors per chip and superlinear scaling of the processor clock with technology generation [1]. However, technology scaling together with frequency and complexity increase result in a significant increase of the power density. This trend, which is becoming a key-limiting factor to the performance of current state-of-the-art microprocessors [2–5], is likely to continue in future generations as well [4, 6]. The higher power density leads to increased heat dissipation and consequently higher operating temperature [7, 8].

To handle higher operating temperatures, chip manufacturers have been using more efficient and more expensive cooling solutions [6, 9]. While such solutions were adequate in the past, these packages are now becoming prohibitively expensive, as the relationship between cooling capabilities and cooling costs is not linear [4, 6]. To reduce packaging cost, current processors are usually designed to sustain the thermal requirement of typical workloads and utilize dynamic thermal management (DTM) techniques when temperature exceeds the design-set point [4, 10]. When the operating temperature reaches a predefined threshold, the DTM techniques reduce the processor’s power consumption in order to allow it to cool down [4, 6, 7, 11–13]. An example of such a DTM mechanism is to reduce the consumed power through duty-cycle-based throttling. While it is very effective achieving its goal, each DTM event comes with a significant performance penalty [4, 7].

Moreover, the reliability of electronic devices and therefore of microprocessors depends exponentially on the operation temperature [4, 5, 14–17]. Viswanath et al. [5] note that even small differences in operating temperature, in the order of 10°C–15°C, can result in a 2x difference in the lifespan of the devices.

Finally, higher temperature leads to power and energy inefficiencies mainly due to the exponential dependence of leakage power on temperature [4, 6, 7, 13]. As in future generations, leakage current is expected to consume about 50% of the total power [1, 3] this issue will become more serious. Additionally, the higher the operating temperature is, the more aggressive the cooling solution must be (e.g., higher
fan speeds) which will lead to further increase in power consumption [11, 12].

The chip multiprocessors (CMP) architecture has been proposed by Olukotun et al. [2] as a solution able to extend the performance improvement rate without further complexity increase. The benefits resulting from this architecture are proved by the large number of commercial products that adopted it, such as IBM’s Power 5 [18], SUN’s Niagara [19], Intel’s Pentium-D [20], and AMD’s Athlon 64 X2 [21].

Recently, CMPs have been successfully used for multimedia applications as they prove able to offer significant speedup for these types of workload [22–24]. At the same time, embedded devices have an increasing demand for multiprocessor solutions. Goodacre [25] states that 3 G handsets are proved by the large number of commercial products that may result in performance degradation. Additionally, the characteristics of the CMP, that is, multiple cores packed together, enable execution scenarios that can cause excessive thermal stress and significant performance penalties.

To address these problems, we propose thermal-aware scheduling. Specifically, when scheduling a process for execution, the operating system determines on which core the process will run based on the thermal state of each core, that is, its temperature and cooling efficiency. Thermal-aware scheduling is a mechanism that aims to avoid situations such as creation of large hotspots and thermal violations, which may result in performance degradation. Additionally, the proposed scheme offers opportunities for performance improvements arising not only from the reduction of the number of DTM events but also from enabling per-core frequency increase, which benefits significantly single-threaded applications [10, 28]. Thermal-aware scheduling can be implemented purely at the operating system level by adding the proper functionality into the scheduler of the OS kernel.

The contributions of this paper are the identification of the thermal issues that arise from the technological evolution of the CMP chips, as well as the proposal and evaluation of a thermal-aware scheduling algorithm with two optimizations: thermal threshold and neighborhood awareness. To evaluate the proposed techniques, we used the TISC simulator [29]. The experimental results for future CMP chip configurations showed that simple thermal-aware scheduling algorithms may result in significant performance degradation as the temperature of the cores often reach the maximum allowed value, consequently triggering DTM events. The addition of a thermal threshold results in a significant reduction of DTM events and consequently in better performance. By making the algorithm aware of the neighboring core thermal characteristics (neighborhood aware), the scheduler is able to take better decisions and therefore provide a more stable performance comparing to the other two algorithms.

The rest of this paper is organized as follows. Section 2 discusses the relevant related work, Section 3 presents the most important temperature-induced problems and analyzes the effect they are likely to have on future chip multiprocessors. Section 4 presents the proposed thermal-aware scheduling algorithms. Section 5 describes the experimental setup and Section 6 the experimental results. Finally, Section 7 presents the conclusions to the work.

2. RELATED WORK

As temperature increase is directly related to the consumed power, techniques that aim to decrease the power consumption achieve temperature reduction as well. Different techniques, however, target power consumption at different levels.

Circuit-level techniques mainly optimize the physical, transistor, and layout design [30, 31]. A common technique uses different transistor types for different units of the chip. The architectural-level techniques take advantage of the application characteristics to enable on-chip units to consume less power. Examples of such techniques include hardware reconfiguration and adaptation [32], clock gating and modification of the execution process, such as speculation control [33]. At the application level, power reduction is mainly achieved during the compilation process using specially developed compilers. What these compilers try to do is to apply power-aware optimizations during the application’s optimization phase such as strength reduction and partial redundancy elimination.

Another solution proposed to deal with the thermal issues is thermal-aware floorplanning [34]. The rationale behind this technique is placing hot parts of the chip in locations having more efficient cooling while avoiding the placement of such parts adjacent to each other.

To handle situations of excessive heat dissipation, special dynamic thermal management (DTM) techniques have been developed. Skadron et al. in [4] present and evaluate the most important DTM techniques, dynamic voltage and frequency scaling (DVFS), units toggling and execution migration. DVFS decreases the power consumed by the microprocessor’s chip by decreasing its operating voltage and frequency. As power consumption is known to have a cubic relationship with the operating frequency [35], scaling it down leads to decreased power consumption and consequently decreased heat dissipation. Although very effective in achieving its goal, DVFS introduces significant performance penalty, which is related to the lower performance due to the decreased frequency and the overhead of the reconfiguration event.

Toggling execution units [4], such as fetch engine toggling, targets power consumption decrease indirectly. Specifically, such techniques try to decrease the number of instructions on-the-fly in order to limit the consumed power and consequently allow the chip to cool. The performance penalty comes from the underutilization of the available resources.
Execution migration [13] is another technique targeting thermal issues and maybe the only one from those mentioned above, that does it directly and not through reducing power consumption. When a unit gets too hot, execution is migrated to another unit that is able to perform the same operation. For this migration to be possible, replicated and idle units must exist.

Executing a workload in a thermal-aware manner has been proposed by Moore ef al. [12] for large data-centers. Specifically, the placement of applications is such that servers executing intensive applications are in positions favored by the cold-air flow from the air conditioners. Thermal-aware scheduling follows the same principles but applies this technique to CMPs.

Donald and Martonosi [36] present a thorough analysis of thermal management techniques for multicore architectures. They classify the techniques they use in terms of core throttling policy, which is applied locally to a core or to the processor as a whole, and process migration policies. The authors concluded that there is significant room for improvement.

3. CMP THERMAL ISSUES

The increasing number of transistors that technology advancements provide, will allow future chip multiprocessors to include a larger number of cores [26]. At the same time, as technology feature size shrinks, the chip’s area will decrease. This section examines the effect these evolution trends will have on the temperature of the CMP chip. We start by presenting the heat transfer model that applies to CMPs and then discuss the two evolution scenarios: smaller chips and more cores on the same chip.

3.1. Heat transfer model in CMPs

Cooling in electronic chips is achieved through heat transfer to the package and consequently to the ambient, mainly through the vertical path (Figure 1(a)). At the same time, there is heat transfer between the several units of the chip and from the units to the ambient through the lateral path. In chip multiprocessors, there is heat exchange not only between the units within a core but also across the cores that co-exist on the chip (Figure 1(b)). As such, the heat produced by each core affects not only its own temperature but also the temperature of all other cores.

The single chip microprocessor of Figure 1(a), can emit heat to the ambient from all its 6 cross-sectional areas whereas each core of the 4-core CMP (Figure 1(b)) can emit heat from only 4. The other two cross-sectional areas neighbor to other cores and cooling through that direction is feasible only if the neighboring core is cooler. Even if the temperature of the neighboring core is equal to that of the ambient, such heat exchange will be poor when compared to direct heat dissipation to the ambient due to the low thermal resistivity of silicon [4]. Furthermore, as the number of on-chip cores increases, there will be cores with only 2 “free” edges (cross-sectional areas at the edge of the chip), further reducing the per-core cooling ability (Figure 1(c)). Finally, if the chip’s area does not change proportionally, the per-core “free” cross-sectional area will reduce harming again the cooling efficiency. All the above lead us to conclude that CMPs are likely to suffer from higher temperature stress compared to single chip microprocessor architectures.

3.2. CMP evolution trends

3.2.1. Trend 1: decreasing the chip size

As mentioned earlier, technology improvements and feature size shrink will allow the trend of decreasing chip’s size to continue. This chip’s area decrease results in higher operating temperature as the ability of the chip to cool by vertically dissipating heat to the ambient is directly related to its area (Section 3.1). As such, the smaller the chip size is, the less efficient this cooling mechanism is. The most important consequence of higher operating temperature is the significant performance penalty caused by the increase of DTM events. Further details about this trend are presented in Section 6.1.

3.2.2. Trend 2: increasing the number of cores

As the number of on-chip core increases, so does the throughput offered by the CMP. However, if the size of the chip does not scale, the per-core area will decrease. As shown previously in Section 3.2, this has a negative effect on the operating temperature and consequently on the performance of the multiprocessor. A detailed study about the effect of increasing the number of on-chip cores will be presented in Section 6.1 together with the experimental results.

3.3. Reliability

Adding more cores to the chip improves the fault tolerance by enabling the operation of the multiprocessor with the remainder cores. Specifically, a CMP with 16 cores can be made to operate with 15 cores if one fails.

More cores on the chip, however, will decrease the chip-wide reliability in two ways. The first is justified by the characteristics of failure mechanisms. According to the sum-of-failure-rates (SOFR) model [37, 38], the failure rate of a CMP can be modeled as a function of the failure rate of its basic core ($\lambda_{BC}$) as shown by (1). In this equation, $n$ is the number of on-chip cores, all of which are assumed to have the same failure rate ($\lambda_i^{BC} = \lambda^{BC}$ $\forall i$). Even if we neglect failures due to the interconnects, the CMP chip has $n$-times greater failure rate compared to its Basic Core,

$$\lambda_{\text{CMP}} = \sum_{i=1}^{n} (\lambda_i^{BC}) + \lambda_{\text{Interconnects}} = n \cdot \lambda^{BC} + \lambda_{\text{Interconnects}}.$$

The second way, more cores on the chip affect chip-wide reliability is related to the fact that higher temperatures exponentially decrease the lifetime of electronic devices [4, 5, 14–17]. As we have shown in Section 3.2, large-scale
CMPs will suffer from larger thermal stress, accelerating these temperature-related failure mechanisms.

It is also necessary to mention that other factors that affect the reliability are the Spatial (different cores having different temperatures at the same time point) and temporal (differences in the temperature of a core over the time) temperature diversities.

3.4. Thermal-aware floorplanning

Thermal-aware floorplanning is an effective widely used technique for moderating temperature-related problems [17, 34, 39, 40]. The rationale behind it is placing hot parts of the chip in locations having more efficient cooling while avoiding the placement of such parts adjacent to each other.

However, thermal-aware floorplanning is likely to be less efficient when applied to CMPs as core-wide optimal decisions will not necessarily be optimal when several cores are packed on the same chip. Referring to Figure 2(d), although cores A and F are identical, their thermally optimal floorplan is likely to be different due to the thermally different positions they have on the CMP. These differences in the optimal floorplan are likely to increase as the number of on-chip cores increases due to the fact that the number of thermally different locations increase with the number of on-chip cores. Specifically, as Figures 2(a) to 2(d) show, for a CMP with $n^2$ identical square cores, there will be $(\lceil n/2 \rceil \cdot (\lceil n/2 \rceil + 1))/2$ different locations.

4. THERMAL-AWARE SCHEDULING

4.1. Scheduling

At any given time point, the operating system’s ready list contains processes waiting for execution. At the same time, each core of the CMP may be either idle or busy executing a process (Figure 3). If idle cores exist, the operating system must select the one on which the next process will be executed.

4.2. The ideal operation scenario

In the ideal case, each core has a constant temperature since the processor was powered-on and therefore no temporal temperature diversities exist. Additionally, this temperature is the same among all cores eliminating spatial temperature diversities. The decrease of spatial and temporal temperature diversities will have a positive effect on chip’s reliability. Of course, this common operating temperature should be as low as possible for lower power consumption, less need for cooling, increased reliability, and increased performance. Finally, the utilization of each core, that is, the fraction of time a core is nonidle should be the same in order to avoid cases where a core has “consumed its lifetime” whereas others have been active for very short. Equal usage should also take into account the thermal stress caused to each core by the applications it executes. Specifically, the situation where a core has mainly being executing temperature intensive applications whereas others have mainly been executing moderate or low stress applications is unwanted. Equal usage among cores will result in improving the chip-wide reliability.

4.3. Highly unwanted scenarios

Several application-execution scenarios that can lead to highly unwanted cases, such as, large performance penalties
or high thermal stress are discussed in this section. These scenarios do not necessarily describe the worse case, but are presented to show that temperature unaware scheduling can lead to situations far from the ideal with consequences opposite to those presented above. Simple thermal-aware scheduling heuristics are shown to prevent such cases.

### 4.3.1. Scenario 1: large performance loss

As mentioned earlier, the most direct way the processor’s temperature can affect its performance is due to more frequent activation of DTM events, which occur each time the temperature of the core exceeds a predefined threshold. The higher the initial temperature of the core is, the easier it is to reach this predefined threshold is. For the temperature of a core to rise, its own heat generation (local) must be larger than the heat it can dissipate to the ambient and to the neighboring cores. However, a core can only dissipate heat to its neighbors if they are cooler. The local heat generation is mainly determined by the application running on the core which may be classified as “hot,” “moderate,” and “cool” [4, 10, 34] depending on the heat it generates. Therefore, the worse case for large loss of performance is to execute a hot process on a hot core that resides in a hot neighborhood.

Let us assume that the CMP’s thermal snapshot (the current temperature of its cores) is the one depicted in Figure 4(a), and that a hot process is to be scheduled for execution. Four cores are idle and thus candidate for executing the new process: C3, D4, E3, and E4. Although C3 is the coolest core, it is the choice that will cause the largest performance loss. C3 has reduced cooling ability due to being surrounded by hot neighbors (C2, C4, B3, and D3) and due to not having free edges, that is, edges of the chip. As such, its temperature will reach the threshold soon and consequently activate a DTM event, leading to a performance penalty.

A thermal-aware scheduler could identify the inappropriateness of C3 and notice that although E4 is not the coolest idle core of the chip, it has two advantages: it resides in a rather cool area and neighbors to the edge of the chip both of which enhance its cooling ability. It would prefer E4 compared to E3 as E4 has two idle neighbors and compared to D4 as it is cooler and has more efficient cooling.

### 4.3.2. Scenario 2: hotspot creation

The “best” way to create a hotspot, that is, an area on the chip with very high thermal stress is to force very high temperature on adjacent cores. This could be the result of running hot applications on the cores and at the same time reducing their cooling ability.

Such a case would occur if a hot application was executed on core E3 of the CMP depicted in Figure 4(b). This would decrease the cooling ability of its already very hot neighbors (E2, E4, and D3). Furthermore, given that E3 is executing a hot application and that it does not have any cooler neighbor, it is likely to suffer from high temperature, soon leading to the creation of a large hotspot at the bottom of the chip.

A thermal-aware scheduler would take into account the impact such a scheduling decision would have, not only on the core under evaluation but also on the other cores of the chip, thus avoiding such a scenario.

### 4.3.3. Scenario 3: high spatial diversity

The largest spatial diversities over the chip appear when the temperature of adjacent cores differ considerably. Chess like scheduling (Figure 5) is the worse case scenario for spatial diversities as between each pair of busy and probably hot cores an idle, thus cooler, one exists.

A thermal-aware scheduler would recognize this situation, as it is aware of the temperature of each core, and moderate the spatial diversities.

### 4.3.4. Scenario 4: high temporal diversity

A core will suffer from high temporal diversities when the workload it executes during consecutive intervals has opposite thermal behavior. Let us assume that the workload
Figure 5: Chess-like scheduling and its effect on spatial temperature diversity. The chart shows the trend temperature is likely to follow over the lines shown on the CMP.

Figure 4: Thermal-aware scheduling and its effect on spatial temperature diversity. The chart shows the trend temperature is likely to follow over the lines shown on the CMP.

4.4. Thermal-aware scheduling on chip multiprocessors

Thermal-Aware Scheduling (TAS) [27] is a mechanism that aims to moderate or even eliminate the thermal-induced problems of CMPs presented in the previous section. Specifically, when scheduling a process for execution, TAS selects one of the available cores based on the core’s “thermal state,” that is, its temperature and cooling efficiency. TAS aims at improving the performance and thermal profile of the CMP, by reducing its temperature and consequently avoiding thermal violation events.

4.4.1. TAS implementation or a real OS

Implementing the proposed scheme at the operating system level enables commodity CMPs to benefit from TAS without any need for microarchitectural changes. The need for scheduling is inherent in multiprocessors operating systems and therefore, adding thermal awareness to it, by enhancing its kernel, will cause only negligible overhead for schedulers of reasonable complexity. The only requirement is an architecturally visible temperature sensor for each core, something rather trivial given that the Power 5 processor [18] already embeds 24 such sensors. Modern operating systems already provide functionality for accessing these sensors through the advanced configuration and power interface (ACPI) [41]. The overhead for accessing these sensors is minimal and so we have not considered it in our experimental results.

4.4.2. Thermal-aware schedulers

In general, a thermal-aware scheduler, in addition to the core’s availability takes into account its temperature and other information regarding its cooling efficiency.

Although knowing the thermal type of the workload to be executed can increase the efficiency of TAS, schedulers that operate without this knowledge, as those presented below, are shown by our experimental results to provide significant benefits. Our study is currently limited to simple, stateless scheduling algorithms which are presented next.

Coolest

The new process is assigned to the Coolest idle core. This is the simplest thermal-aware algorithm and the easiest to implement.

Neighborhood

This algorithm calculates for each available core a cost function (equation (2)) and selects the core that minimizes it. This cost function takes into consideration the following:

(i) temperature of the candidate core ($T_c$),
(ii) average temperature of directly neighboring cores ($\overline{T_{DA}}$),
(iii) average temperature of diagonally neighboring cores ($\overline{T_{dA}}$),
(iv) number of nonbusy directly neighboring cores ($\overline{NB_{DA}}$),
(v) the number of “free” edges of the candidate core ($N_{fe}$).

Each parameter is given a different importance through the $a_i$ weights. The value of these weights is determined statistically through experimentation in order to match the characteristics of the CMP. The rationale behind this algorithm is that, the lower the temperature of the core’s neighborhood is, the easier it will be to keep its temperature at low levels due to the intercore heat exchange. Cores neighboring with the edge of the chip are beneficial due to the increased heat abdution rate from the ambient.

$$\text{Cost} = a_1 \cdot T_c + a_2 \cdot \overline{T_{DA}} + a_3 \cdot \overline{T_{dA}} + a_4 \cdot \overline{NB_{DA}} + a_5 \cdot \overline{N_{fe}}.$$  \hspace{1cm} (2)

Threshold neighborhood

The Threshold Neighborhood algorithm uses the same cost function as the Neighborhood algorithm, but schedules a process for execution only if a good enough core exist. This good enough threshold is a parameter of the algorithm. A core is considered appropriate if its cost function is lower than this...
threshold (in contrast, when the neighborhood algorithm is used, a process is scheduled no matter the value of the cost function). This algorithm is nongreedy as it avoids scheduling a process for execution on a core that is available but in a thermally adverse state.

Although one would expect that the resulting underutilization of the cores could lead to performance degradation, the experimental results showed that with careful tuning, performance is improved due to the reduction of the number of DTM events.

**MST heuristic**

The maximum scheduling temperature (MST) heuristic, is not an algorithm itself but an option that can be used in combination with any of the previously mentioned algorithms. Specifically, MST prohibits scheduling a process for execution on idle cores when their temperature is higher than a predefined threshold ($MST_T$).

5. **EXPERIMENTAL SETUP**

To analyze the effect of thermal problems on the evolution of the CMP architecture and to quantify the potential of TAS in solving these issues, we conducted several experiments using a specially developed simulator.

5.1. **The simulated environment**

At any given point in time, the operating system’s ready list contains processes ready to be executed. At the same time, each core of the CMP may be either busy executing a process or idle. If idle cores exist, the operating system, using a scheduling algorithm selects one such core and schedules on it a process from the ready list. During the execution of the simulation, new processes are inserted into the ready list and wait for their execution. When a process completes its execution, it is removed from the execution core, which is thereafter deemed as idle.

The heat produced during the operation of the CMP and the characteristics of the chip define the temperature of each core. For the simulated environment, the DTM mechanism used is that of process migration. As such, when the temperature of a core reaches a predefined threshold ($45 \degree C$ above the ambient), the process it executes is “migrated” to another core. Each such migration event comes with a penalty (migration penalty — $DTM-P$), which models the overheads and performance loss it causes (e.g., invocation of the operating system and cold caches effect).

5.2. **The simulator**

The simulator used is the Thermal Scheduling Simulator for Chip Multiprocessors (TSIC) [29], which has been developed specially to study thermal-aware scheduling on chip multiprocessors. TSIC models CMPs with different number of cores whereas it enables studies exploring several other parameters, such as the maximum allowed chip temperature, chip utilization, chip size, migration events, and scheduling algorithms.

5.2.1. **Process model**

The workload to be executed is the primary input for the simulator. It consists of a number of power traces, each one modeling one process. Each point in a power trace represents the average power consumption of that process during the corresponding execution interval. Note that all intervals have the same length in time. As the power consumption of a process varies during its execution, a power trace is likely to consist of different power consumption values for each point. The lifetime of a process, that is, the total number of simulation intervals that it needs to complete its execution, is defined as the number of points in that power trace.

TSIC loads the workload to be executed in a workload list and dynamically schedules each process to the available cores. When the temperature of a core reaches a critical point ($DTM$-threshold), the process running on it must be either migrated to another core or suspended to allow the core to cool. Such an event is called thermal violation event. If no cores are available, that is, they are all busy or do not satisfy the criteria for the MST heuristic of Threshold Neighborhood algorithm, the process is moved back to the workload list and will be rescheduled when a core becomes available.
Each time a process is to be assigned for execution, a scheduling algorithm is invoked to select a core, among the available ones, to which the process will be assigned for execution.

For the experiments presented in this paper, the workload used consists of 2500 synthetic randomly produced processes with average lifetime equal to 100 simulation intervals (1 millisecond per interval) and average power consumption equal to 10 W. The rationale behind using a short average lifetime is to model the OS’s context-switch operation. Specifically, each simulated process is to be considered as the part of a real-world process during two consecutive context switches.

5.2.2. The chip multiprocessor

TSIC uses a rather simplistic model for the chip’s floorplan of the CMP. As depicted in Figure 7, each core is considered to cover a square area whereas the number of cores on the chip is always equal to $n^2$ where $n$ is the number of cores in each dimension. In current TSIC implementation, cores are assumed to be areas of uniform power consumption. The area of the simulated chip is equal to 256 mm$^2$ (the default of the Hotspot simulator [4]).

5.2.3. Thermal model

TSIC uses the thermal model of Hotspot [4] which has been ported into the simulator. The floorplan is defined by the number of cores and the size of the chip.

5.2.4. Metrics

During the execution of the workload, TSIC calculates the total number of intervals required for its execution ($Cycles$), the number of migrations ($Migrations$) as well as several temperature-related statistics listed below.

(i) Average Temperature: the Average Temperature represents the average temperature of all the cores of the chip during the whole simulation period. The Average Temperature is given by (3), where $T_{t,i,j}$ is the temperature of core $i,j$ during simulation interval $t$, $S_T$ is the total number of simulation intervals, and $n$ is the number of cores.

$$\text{Average Temperature} = \bar{T} = \frac{S_T}{n \cdot S_T} \sum_{t=0}^{S_T-1} \left( \frac{1}{n^2} \cdot \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (T_{t,i,j}) \right).$$

(ii) Average Spatial Diversity: the Spatial Diversity shows the variation in the temperature among the cores at a given time. The Average Spatial Diversity (equation (4)) is the average of the Spatial Diversity during the simulation period. A value equal to zero means that all cores of the chip have the same temperature at the same time, but possibly different temperature at different points in time. The larger this value is, the greater the variability is. In the Average Spatial Diversity equation, $T_{t,i,j}$ is the temperature of core $i,j$ during simulation interval $t$, $S_T$ is the total number of simulation intervals, and $n$ is the number of cores.

$$\text{Average Spatial Diversity} = \sum_{t=0}^{S_T-1} \left[ \frac{1}{n^2 \cdot S_T} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |T_{t,i,j} - \bar{T}| \right].$$

(iii) Average Temporal Diversity: the Average Temporal Diversity is a metric of the variation of the average chip temperature, across all cores, and is defined by (5). In the Average Temporal Diversity equation $T_{t,i,j}$ is the temperature of core $i,j$ during simulation interval $t$, $\bar{T}^t = 1/n^2 \cdot \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} T_{t,i,j}$

$$\text{Average Temporal Diversity} = \sum_{t=0}^{S_T-1} \left[ \frac{1}{n \cdot S_T} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \left| T_{t,i,j} - \bar{T}^t \right| \right].$$
is the average chip temperature during simulation interval \( t \), \( T \) is the average chip temperature as defined by (3), \( S_T \) is the total number of simulation intervals, and \( n \) is the number of cores,

\[
\text{Average Temporal Diversity} = \frac{\sum_{j=0}^{S_T} \left| T_j - T \right|}{n \cdot S_T}.
\]

(5)

(iv) **Efficiency:** efficiency is a metric of the actual performance the multiprocessor achieves in the presence of thermal problems compared to the potential offered by the CMP. Efficiency is defined by (6) as the ratio between the time required for the execution of the workload (Workload Execution Time) and the time it would require if no thermal violation events existed (Potential Execution Time, (7)). The maximum value for the Efficiency metric is 1 and represents full utilization of the available resources,

\[
\text{Efficiency} = \frac{\text{Potential Execution Time}}{\text{Workload Execution Time}}.
\]

(6)

Potential Execution Time = \( \sum_{n=1}^{\text{# processes}} \frac{\text{Lifetime (Process}_n\text{)}}{\text{Number of Cores}} \).

(7)

5.2.5. **Scheduling algorithms**

For the experimental results presented in Section 6, all threshold values for the scheduling algorithms, the \( a_i \) factors in (2), the MST-T, and the “Threshold Neighborhood,” have been statically determined through experimentation. Although adaptation of these threshold values could be done dynamically, this would result in an overhead for the scheduler of the operating system. We are however currently studying these issues.

6. **RESULTS**

6.1. **Thermal behavior and its implications for future CMPs**

In this section we present the thermal behavior and its impact on the performance for future CMP configurations which are based on the technology evolution. This leads to chips of decreasing area and/or more cores per chip. For the results presented, we assumed that the CMPs are running an operating system that supports a minimal overhead thermal scheduling algorithm such as Coolest (baseline algorithm for this study). Consequently these results are also an indication of the applicability of simple thermal scheduling policies.

6.1.1. **Trend 1: decreasing the chip size**

As mentioned earlier, technology improvements and feature size shrink will allow the trend of decreasing the chip size to continue. Figure 8(a) depicts the effect of this chip size decrease while keeping the consumed power constant for a CMP with 16 cores. The results clearly show the negative effect of chip’s area decrease on the average temperature and the efficiency of the multiprocessor. This is explained by the fact that the ability of the chip to cool by vertically dissipating heat to the ambient is directly related to its area (Section 3.1).

Lower cooling ability leads to higher temperature, which in turn leads to increased number of migrations, and consequently to significant performance loss. The reason for which the temperature only asymptotically approximates 45°C is related to the protection mechanism used (process migration) which is triggered at 45°C. Notice that the area of typical chips today does not exceed 256 mm², which is the point beyond which it is possible to observe considerable performance degradation. A migration penalty (DTM-P) of one interval is used for these experiments. This value is small compared to what would apply in a real world system and consequently these charts present an optimistic scenario.

Another unwanted effect is related to the spatial and temporal diversities, which also become worse for smaller chips (Figure 8(b)) and is justified mainly by the higher operating temperatures. Notice that in this chart we limit the chip size range to that for which no migrations exist in order to exclude from the trend line the effect of migrations.

6.1.2. **Trend 2: increasing the number of cores**

As explained in Section 3.2, due to thermal limitations, the throughput potential offered by the increased number of cores cannot be exploited unless the size of the CMP is scaled proportionally. Figure 9 depicts the efficiency and

![Figure 8: (a) Efficiency and temperature (°C above the ambient) and (b) spatial and temporal diversities for different chip sizes.](image-url)
temperature for CMPs with different number of cores (4, 16, 36, and 64) for three different utilization points (50%, 80%, and 100%). Utilization shows the average fraction of cores that are active at any time point and models the execution stress of the multiprocessor.

The efficiency of the different CMP configurations studied is depicted in Figure 9(a). The decrease in efficiency with the increase in the number of on-chip cores is justified by the decrease in the per-core area and consequently of the vertical cooling capability. The increased utilization also decreases the cooling capabilities of cores but this is related to the lateral heat transfer path. Specifically, if a neighboring core is busy, and thus most likely hot, cooling through that direction is less efficient. In the worse scenario, a core will receive heat from its neighbors and instead of cooling, it will get hotter. Both factors have a negative effect on temperature (Figure 9(b)) and consequently in the number of migration events, which is the main reason for performance loss. It is relevant to notice that for the 36- and 64-core CMPs the average temperature is limited by the maximum allowed threshold, which has been set to 45°C for these experiments.

The workload execution time for the different CMP configurations studied is depicted in Figure 9(c). For the 4-core CMP, higher utilization leads to a near proportional speedup, which is significantly smaller for the 16-core CMP and almost diminishes for multiprocessors with more cores. This indicates the constrain thermal issues pose on the scalability offered by the CMP’s architecture. It is relevant to notice that for the 100% utilization point, the 64-core chip has almost the same performance as the 16-core CMP. This behavior is justified by the large number of migration events suffered by the large scale CMPs.

Figure 9(d) displays the slowdown of each configuration due to temperature related issues taking the utilization into consideration, that is, if a configuration with utilization 50% executes the workload in 2X cycles where the same configuration with 100% utilization executes it in X cycles, the former is considered to have zero slowdown. The results emphasize the limitations posed by temperature issues on fully utilizing the available resources. Notice that these limitations worsen as the available resources increase.

Finally, Figure 10 depicts the spatial and temporal diversities of the CMP configurations studied, when utilization is equal to 100%. Both diversities are shown to worsen when more cores coexist on the chip. This is not only due to the higher temperature but also due to variability caused by the larger number of on-chip cores.

### 6.2. Optimization 1: thermal threshold

The results from the previous section showed a significant drop in performance as the maximum operating temperature
is reached. To avoid this performance degradation, we propose to enhance the basic Thermal-aware scheduling policy (Coolest) by using a threshold on the core’s temperature. This is what we named the Coolest + MST scheduling scheme, that is, a process is executed on the coolest available core only if a core with temperature lower than $N^\circ C$ exists. In our case, we use 40$^\circ C$ as the threshold value (MST-T), that is, five degrees lower than the maximum allowed temperature. The goal for these experiments is to show how, Coolest + MST, is able to improve the performance by reducing the number of migrations. In addition, we set the DTM-Penalty to zero, which is the reason why we will not present performance results.

Table 1 presents the number of migrations and the average temperature for the execution scenarios mentioned before. As can be seen from the results, the Coolest + MST heuristic is able to significantly decrease the number of migration events. The potential of this algorithm increases with the number of cores. This is a first-class indication that performance improvement can be achieved. At the same time, this TAS scheme decreases the average chip temperature by approximately 2$^\circ C$ for the 16-core and 2.5$^\circ C$ for the 25-core CMP.

Figure 11 depicts the number of migrations and temperature of CMPs with different number of cores as the MST-Threshold (MST-T) ranges from 40$^\circ C$ to 45$^\circ C$. Note that when MST-T is equal to the DTM-threshold (DTM-T) (45$^\circ C$), scheduling is the same as what would apply without MST (Coolest).

As depicted in Figure 11(a) for both the 16-core and 25-core CMP, the number of migrations increases with the MST-T. This is due to the fact that cores with very high temperature are allowed to be used. The same trend stands for the average temperature of the chip (Figure 11(b)), which justifies what is observed for migrations.

No performance results are presented for this experiment as the DTM-Penalty value has been set to be equal to 1 interval only. As such, its impact on performance is minimal. However, as mentioned earlier, in a real-world system the DTM-Penalty will be significantly larger.

When DTM events are penalized, execution using the Coolest policy may not complete. If the scheduling algorithm
is greedy in that it tries to fully utilize the available resources no matter their thermal state, a scenario described by a *vicious-circle* of continued process-migrations is possible. Such a scenario appears when cores with very high temperature are used and at the same time, the average temperature of the chip is close to DTM-T. For example, this scenario happens when executing the experimental workload on a 36-core CMP with *Coolest*.

### 6.3. Optimization 2: neighborhood awareness

The results from the previous section showed that adding a *Threshold* to the simple thermal-aware scheduling (*Coolest*) policy can significantly decrease the number of migration events. Nevertheless, the *Coolest + MST* algorithm uses local information to make the scheduling decisions, that is, it considers only the temperature of the candidate cores.

In this section, we present the results for an algorithm that takes into consideration not only the temperature of the candidate cores but also the temperatures of all the surrounding or neighboring cores. We presented previously, in Section 4.4, two algorithms that use this information, the *Neighborhood* and the *Threshold Neighborhood*. The results presented in this section are for the *Threshold Neighborhood* as its performance is much better compared to the simple *Neighborhood*.

As we have not yet completely tuned the *Threshold Neighborhood* algorithm, we present results only for a single setup of a CMP with 16 cores and 100% utilization. Figure 12(a) depicts the number of migrations for the two algorithms under evaluation (*Threshold Neighborhood* and *Coolest + MST*), for different DTM-Penalties.

The number of migration events suffered by the *Threshold Neighborhood* algorithm is always smaller compared to those of the *Coolest + MST*. For the *Coolest + MST* algorithm, this number of migrations increases with the DTM-Penalty as the additional time required for the execution of the workload worsens the already problematic thermal state of the CMP. On the other hand, for the *Threshold Neighborhood* algorithm the number of migrations decrease with the DTM-Penalty. This shows the ability of the algorithm to adapt to the different DTM-Penalty values. Specifically, as this penalty increases, the algorithm becomes more strict when evaluating the thermal appropriateness of a core.

It must be noted here that the parameters of the *Threshold Neighborhood* algorithm are not the same for all situations. As the migration penalty increases, the configuration that performs better is the one that has smaller weight for the temperature of the candidate core and larger weight for the number of nonbusy directly adjacent cores (See (2)). However, when the migration penalty is small, a conservative selection for execution cores is not desired as the effect of migrations is less important.

Figure 12(b) depicts the execution time of the experimental workload for the different scenarios studied. The performance of the *Coolest + MST* algorithm worsens as the DTM Penalty increases mainly due to two reasons. The first is related to the increase of migration events whereas the second to the fact that each migration has a larger cost. In contrast, the performance of *Threshold Neighborhood* algorithm is almost constant. This is due to the ability of the algorithm to decrease the number of migrations it suffers, as their cost increases with the migration penalty.
Finally, Figure 12(c) depicts the average temperature of the chip for the different configurations studied. It is obvious that the Threshold Neighborhood algorithm manages not only to increase performance but also to decrease the temperature of the chip. This was expected, as only when the chip has better temperature characteristics, migration events can be controlled.

This exploration clearly shows that trying to fully utilize the available resources without taking into consideration the thermal issues may significantly affect performance. Another conclusion is that it is often beneficial to be conservative on utilizing the on-chip resources as this will allow better cooling, will decrease the number of migrations, and consequently enhance performance.

6.4. Summary

In the previous sections, we showed the performance improvements that may be achieved by two optimizations to the basic TAS algorithm (Coolest). On the one hand, Coolest + MST uses a threshold to reduce the number of migrations. On the other hand, Threshold Neighborhood uses local and information about the surrounding cores to take the scheduling decisions. In addition to reducing the number of migrations, this algorithm has also the potential to achieve a better chip-wide thermal behavior. A simple comparison between the three TAS algorithms is presented in Figure 13. This Figure depicts the execution time for the three algorithms for different DTM penalty values on a 16-core CMP.

The results in Figure 13 show that Coolest is intolerant to the increase of the DTM penalty, resulting in large performance loss. This is due to the larger number of migrations compared with the other algorithms. The Coolest + MST performs well for smaller values of DTM penalty. Nevertheless, it is possible to observe that the execution time almost doubled for Coolest + MST when the penalty increased from 15 to 20. In contrast with the previous two algorithms, for Threshold Neighborhood the execution time is not affected by the increase in the DTM penalty resulting in almost no performance degradation. As such, we are led to conclude that the Threshold Neighborhood is the most stable TAS algorithm that achieves the best overall results.

7. Conclusions

In this paper, we have shown that packing a large number of cores onto the same chip reduces the per-core cooling ability compared to a single chip microprocessor further increasing the temperature-induced problems. Additionally, we have presented several scenarios that result in excessive thermal stress or significant performance loss due to insufficient heat dissipation. In order to minimize or eliminate these problems, we propose thermal-aware scheduler algorithms that take into account the thermal state of the CMP while assigning processes to cores. We have shown that such a scheduler can decrease or even avoid high-thermal-stress scenarios, at the same time significantly improving the performance.

8. References

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger, “Clock rate versus IPC: the end of the road for conventional microarchitectures,” in Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA ’00), pp. 248–259, ACM Press, Vancouver, BC, Canada, June 2000.
[2] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K. Chang, “The case for a single-chip multiprocessor,” in Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPL ’96), pp. 2–11, ACM Press, Cambridge, Mass, USA, October 1996.
[3] T. Mudge, “Power: a first-class architectural design constraint,” Computer, vol. 34, no. 4, pp. 52–58, 2001.
[4] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, “Temperature-aware microarchitecture: extended discussion and results,” Tech. Rep. TR-CS-2003-08, University of Virginia, Charlottesville, Va, USA, 2003.
[5] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur, “Thermal performance challenges from silicon to systems,” Intel Technology Journal, vol. 4, no. 3, p. 16, 2000.
[6] S. H. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, “Managing the impact of increasing microprocessor power consumption,” Intel Technology Journal, vol. 5, no. 1, p. 9, 2001.
[7] W. Liao and L. He, “Coupled power and thermal simulation with active cooling,” in Proceedings of the 3rd International Workshop on Power-Aware Computer Systems (PACS ’03), vol. 3164 of Lecture Notes in Computer Science, pp. 148–163, San Diego, Calif, USA, December 2003.
[8] A. Dhodapkar, C. H. Lim, G. Cai, and W. R. Daasch, “TEM²P²EST: a thermal enabled multi-model power/performance ESTimator,” in Proceedings of the 1st International Workshop on Power-Aware Computer Systems-Revised Papers (PACS ’00), pp. 112–125, Cambridge, Mass, USA, November 2000.
[9] R. Mahajan, K. Brown, and V. Atluri, “The evolution of microprocessor packaging,” Intel Technology Journal, p. 10, 2001.
[10] Y. Li, D. Brooks, Z. Hu, and K. Skadron, “Performance, energy, and thermal considerations for SMT and CMP architectures,” in Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA ’05), pp. 71–82, San Francisco, Calif, USA, February 2005.

[11] D. Genossar and N. Shamir, “Intel pentium M processor: power estimation, budgeting, optimization and validation,” Intel Technology Journal, vol. 7, no. 2, pp. 44–49, 2003.

[12] J. Mooref, J. Chasef, P. Ranganathanf, and R. Sharmaf, “Making scheduling “cool”: temperature-aware workload placement in data centers,” in Proceedings of the USENIX Annual Technical Conference, pp. 61–75, Anaheim, Calif, USA, April 2005.

[13] S. Heo, K. Barr, and K. Asanović, “Reducing power density through activity migration,” in Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED ’03), pp. 217–222, Seoul, South Korea, August 2003.

[14] Z. Lu, W. Huang, S. Ghosh, J. Lach, M. Stan, and K. Skadron, “Analysis of temporal and spatial temperature gradients for IC reliability,” Tech. Rep. CS-2004-08, University of Virginia, Charlottesville, VA, USA, March 2004.

[15] Z. Lu, J. Lach, M. Stan, and K. Skadron, “Banking chip lifetime: opportunities and implementation,” in Proceedings of the 1st Workshop on High Performance Computing Reliability Issues (HPCRI ’05), San Francisco, Calif, USA, February 2005.

[16] C. Bailey, “Modelling the effect of temperature on product reliability,” in Proceedings of 19th Annual IEEE Semiconductor Thermal Measurement and Management Symposium (SEMITHERM ’03), pp. 324–331, San Jose, Calif, USA, March 2003.

[17] R. Mukherjee, S. O. Memik, and G. Memik, “Temperature-aware resource allocation and binding in high-level synthesis,” in Proceedings of the 42nd Design Automation Conference (DAC ’05), pp. 196–201, ACM Press, Anaheim, Calif, USA, June 2005.

[18] R. Kalla, B. Sinharoy, and J. M. Tendler, “IBM Power5 chip: a dual-core multithreaded processor,” IEEE Micro, vol. 24, no. 2, pp. 40–47, 2004.

[19] P. Kongetira, K. Aingaran, and K. Olukotun, “Niagara: a 32-way multithreaded Sparc processor,” IEEE Micro, vol. 25, no. 2, pp. 21–29, 2005.

[20] Intel, “Intel Pentium D Processor Product Information,” 2006, http://www.intel.com/products/processor/pentium_d/.

[21] AMD, “AMD Athlon 64X2 Dual-Core Processor,” 2006, http://www.amd.com/.

[22] R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes, “The energy efficiency of CMP vs. SMT for multimedia workloads,” in Proceedings of the 18th Annual International Conference on Supercomputing (ICS ’04), pp. 196–206, Saint-Malo, France, June–July 2004.

[23] E. Debes, “Recent changes and future trends in general purpose processor architectures to support image and video applications,” in Proceedings of IEEE International Conference on Image Processing (ICIP ’03), vol. 3, pp. 85–88, Barcelona, Spain, September 2003.

[24] F. Mombers and D. Mlynek, “Multithreaded multimedia processor merging on-chip multiprocessors and distributed vector pipelines,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS ’99), vol. 4, pp. 287–290, Orlando, Fla, USA, May–June 1999.

[25] J. Goodacre, “Understanding the Options for Embedded Multiprocessing,” 2003, TechOnLine: http://www.techonline.com/community/tech_group/soc/tech_paper/29359.

[26] Intel, “Intel Pentium D Processor Product Information,” 2006, http://www.intel.com/products/processor/pentium_d/.

[27] K. Stavrou and P. Trancoso, “Thermal-aware scheduling: a solution for future chip multiprocessors thermal problems,” in Proceedings of the 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD ’06), pp. 123–126, Dubrovnik, Croatia, August-September 2006.

[28] E. Grochowski, R. Ronen, J. Shen, and H. Wang, “Best of both latency and throughput,” in Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD ’04), pp. 236–243, San Jose, Calif, USA, October 2004.

[29] K. Stavrou and P. Trancoso, “TSIC: thermal scheduling simulator for chip multiprocessors,” in Proceedings of the 10th Panhellenic Conference on Informatics (PCI ’05), vol. 3746 of Lecture Notes in Computer Science, pp. 589–599, Volos, Greece, November 2005.

[30] P. I. Pénzes, M. Nyström, and A. J. Martin, “Transistor sizing: how to control the speed and energy consumption of a circuit,” in Proceedings of the 10th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC ’04), vol. 10, pp. 51–61, Crete, Greece, April 2004.

[31] R. Balasubramonian, D. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, “Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures,” in Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO ’00), pp. 245–257, ACM Press, Monterey, Calif, USA, December 2000.

[32] S. Manne, A. Klauzer, and D. Grunwald, “Pipeline gating: speculation control for energy reduction,” in Proceedings of the 25th Annual International Symposium on Computer Architecture (ISCA ’98), pp. 132–141, Barcelona, Spain, June–July 1998.

[33] Y. Han, I. Koren, and C. A. Moritz, “Temperature aware floorplanning,” in Proceedings of the 2nd Workshop on Temperature-Aware Computer Systems (TACS ’05), Madison, Wis, USA, June 2005.

[34] R. McGowen, C. A. Poirier, C. Bostak, et al., “Power and temperature control on a 90-nm Itanium family processor,” IEEE Journal of Solid-State Circuits, vol. 41, no. 1, pp. 229–237, 2006.

[35] J. Donald and M. Martonosi, “Techniques for multiscop thermal management: classification and new exploration,” in Proceedings of the 33rd International Symposium on Computer Architecture (ISCA ’06), pp. 78–88, Boston, Mass, USA, June 2006.

[36] NIST/SEMA TECH, Assessing Product Reliability, Chapter 8, http://www.itl.nist.gov/div898/handbook/.

[37] J. Irwin, “Thermal-aware task allocation and scheduling for embedded systems,” in Proceedings of the Conference on Design, Automation and Test in Europe (DATE ’05), vol. 2, pp. 898–899, IEEE Computer Society, Munich, Germany, March 2005.
[40] P. Rosinger, B. Al-Hashimi, and K. Chakrabarty, “Rapid generation of thermal-safe test schedules,” in Proceedings of the Conference on Design, Automation and Test in Europe (DATE ’05), vol. 2, pp. 840–845, IEEE Computer Society, Munich, Germany, March 2005.

[41] ACPI, "Advanced Configuration and Power Interface," 2006, http://www.acpi.info/.