Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2007, Article ID 28735, 18 pages
doi:10.1155/2007/28735

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.

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].

Figure 1: Example face detection based on colour filtering.

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 $L' = S \times L$, where L is the transistor channel length. Similarly, the voltage V is scaled by a factor U according to $V' = U \times V$.

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 $P \propto CV^2 f\alpha$, where P depends on the capacitive load switched C, the voltage V, the operating frequency f, and the node switching probability α. 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 α, V, L, and f is achieved by (1),

$$P' = P \times S^2 \times U. \tag{1}$$

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,

$$E' = P' \times D = P' \times \frac{1}{f'} \times C. \tag{2}$$

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),

$$\mathrm{EDP}' = P' \times D^2. \tag{3}$$

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 toward the so-called fast exhaustive search strategies that employ conservative SAD estimations or early exit mechanisms [12, 16, 17].

Recently, many ME optimisation approaches have been proposed to tackle memory efficiency. They employ memory data flow optimisation techniques rather than traditional memory banking techniques. This is achieved by a high degree of on-chip memory content reuse, parallel pel information access, and memory access interleaving [13].

The architectures proposed in this paper implement an efficient fast exhaustive block-matching architecture. ME's high computational requirements are addressed by implementing in hardware an early termination mechanism. It improves upon [17] by increasing the probability of cancellation through a macroblock partitioning scheme. The computational load is shared among 2^{2*n} processing elements (PE). This is made possible in our approach by remapping and partitioning the video content by means of pixel subsampling (see Figure 2). Two architectural variations have been designed using 4 PEs (Figure 3) and 16 PEs, respectively. For clarity all the equations, diagrams, and examples provided concentrate on the $4 \times$ PE architecture only, but can be easily extended.

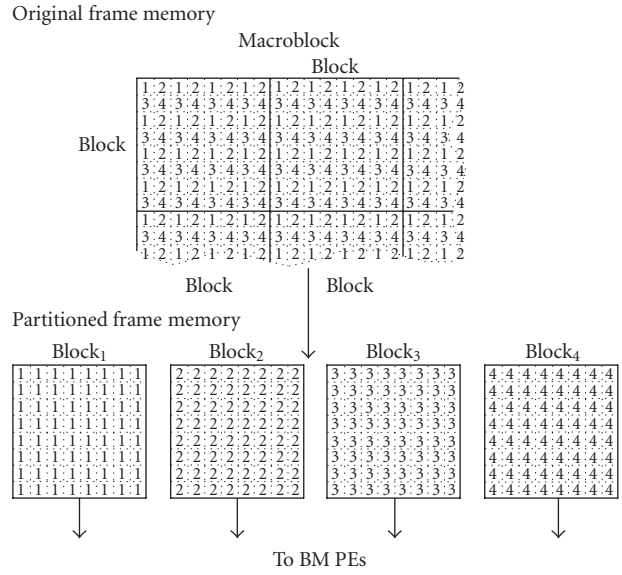


FIGURE 2: Pixel remapping.

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

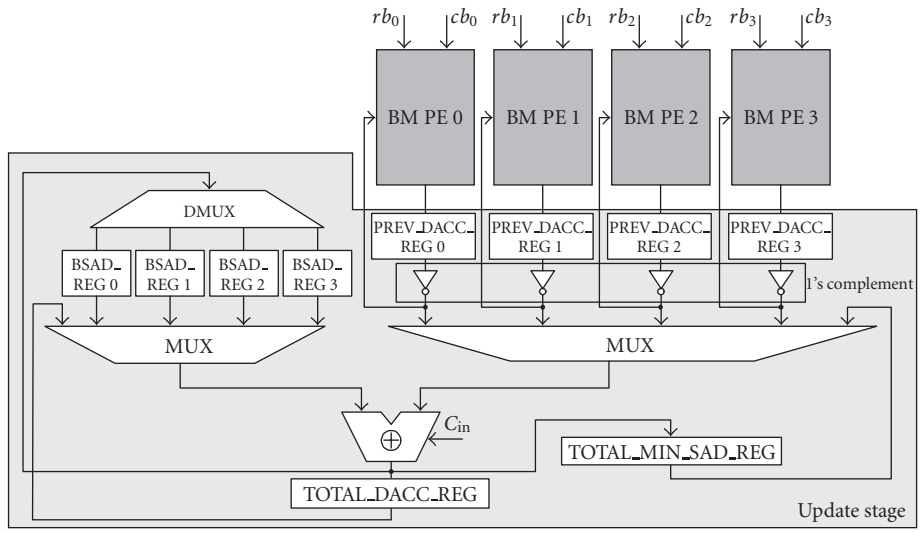


Figure 3: 4 × PE architecture.

block (B) and macroblock (MB) levels. The proposed solution 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 min SAD (7),

$$\begin{aligned}\mathrm{SAD}\,(\,\mathrm{MB}_c,\mathrm{MB}_r\,) &= \sum_{i=1}^{16}\sum_{j=1}^{16}\,\left|\,\mathrm{MB}_c(i,j)-\mathrm{MB}_r(i,j)\,\right| \\ &= \sum_{k=0}^{3}\sum_{i=1}^{8}\sum_{j=1}^{8}\,\left|\,B_{c_k(i,j)-B_{r_k}(i,j)}\,\right| \\ &= \sum_{k=0}^{3}\mathrm{BSAD}_k,\end{aligned} \tag{4}$$

$$\min \mathrm{SAD} = \sum_{k=0}^{3}\min \mathrm{BSAD}_k, \tag{5}$$

$$\mathrm{curr\,SAD}\,(\,\mathrm{MB}_c,\mathrm{MB}_r\,) = \sum_{k=0}^{3}\mathrm{curr\,BSAD}_k, \tag{6}$$

$$\begin{aligned}\mathrm{rel\,SAD}\,(\,\mathrm{MB}_c,\mathrm{MB}_r\,) &= \min \mathrm{SAD} - \mathrm{curr\,SAD}\,(\,\mathrm{MB}_c,\mathrm{MB}_r\,) \\ &= \sum_{k=0}^{3}\,(\,\min \mathrm{BSAD}_k - \mathrm{curr\,BSAD}_k\,).\end{aligned} \tag{7}$$

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,

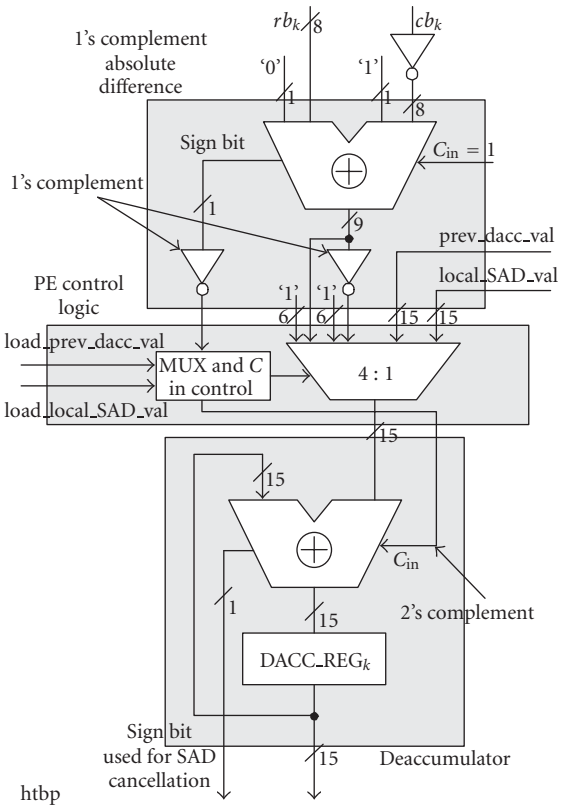


Figure 4: Texture PE.

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$_k$ values are calculated, a de-accumulation function is used instead. The absolute difference is de-accumulated from the DACC_REG$_k$ register (de-accumulator).

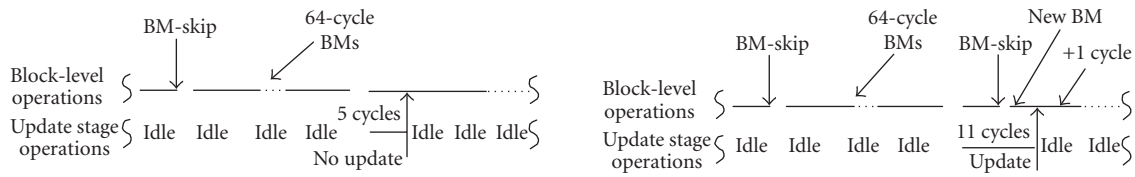
At each moment, the DACC_REG$_k$ stores the appropriate rel BSAD$_k$ value and signals immediately with its sign bit if it becomes negative. The initial value stored in the DACC_REG$_k$ at the beginning of each match is the corresponding min BSAD$_k$ value and is brought through the local_SAD_val inputs. Whenever all the DACC_REG$_k$ 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$_k$ values, stored, respectively, in TOT_MIN_SAD_REG and BSAD_REG$_k$, are updated. Also, the rel BSAD$_k$ corrections, stored beforehand in the PREV_DACC_REG$_k$ registers, have to be made to the PEs' DACC_REG$_k$ registers. The correction operation involves a subtraction of the PREV_DACC_REG$_k$ values (inverters provided in Figure 3 to obtain 2's complement) from the DACC_REG$_k$ 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$_k$ 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 search, 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 $4\times$ 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 $16\times$ 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 $4\times$PE architecture, the $16\times$PE architecture is faster but removes less redundant SAD operations. Thus, choosing between $4 \times$ PE and $16 \times$ 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



(a) Update stage launched but no update

(b) Update stage launched, BM-skip, and update executed

Figure 5: Parallel update stage scenarios.

Table 1: ME architecture comparison for QCIF test sequences.

	SAD operations (millions)					Cycles (millions)				
	1D SA	4 × PE	%	16 × PE	%	1D SA	4 × PE	%	16 × PE	%
Akiyo	1551 M	130 M	8%	357 M	23%	103 M	33 M	31%	22 M	21%
Coastguard	1551 M	443 M	28%	665 M	43%	103 M	110 M	106%	42 M	40%
Foreman	1551 M	509 M	33%	730 M	47%	103 M	127 M	123%	47 M	45%
M&d	1551 M	359 M	23%	603 M	39%	103 M	90 M	86%	40 M	39%
Table tennis	1551 M	408 M	26%	641 M	41%	103 M	102 M	98%	45 M	43%

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 $f_{\max}$ 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 × 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.

Table 2: ME synthesis results and benchmarking.

Architecture	Tech (μm)	Cycle count Max	Min	Average	Gates	PGCC	f (MHz)	Power (mW)	P' (mW)	E' (nJ)	EDP′ (fJs)
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	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

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 Panusopone 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,

$$\text{SAD}\left(B_{\text{cur}}, B_{\text{ref}}\right) = \sum_{i=1}^{i=16} \sum_{j=1}^{j=16} B_{\text{cur}}(i,j) \otimes B_{\text{ref}}(i,j). \quad (8)$$

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):

$$\begin{aligned} \text{TOTAL}_{\text{cur}} &= \text{COMMON} + \text{UNIQUE}_{\text{cur}}, \\ \text{TOTAL}_{\text{ref}} &= \text{COMMON} + \text{UNIQUE}_{\text{ref}}, \end{aligned} \quad (9)$$

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) $\text{UNIQUE}_{\text{cur}}$ is the number of white pixels in the current BAB but not in the reference BAB.
(e) $\text{UNIQUE}_{\text{ref}}$ 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

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

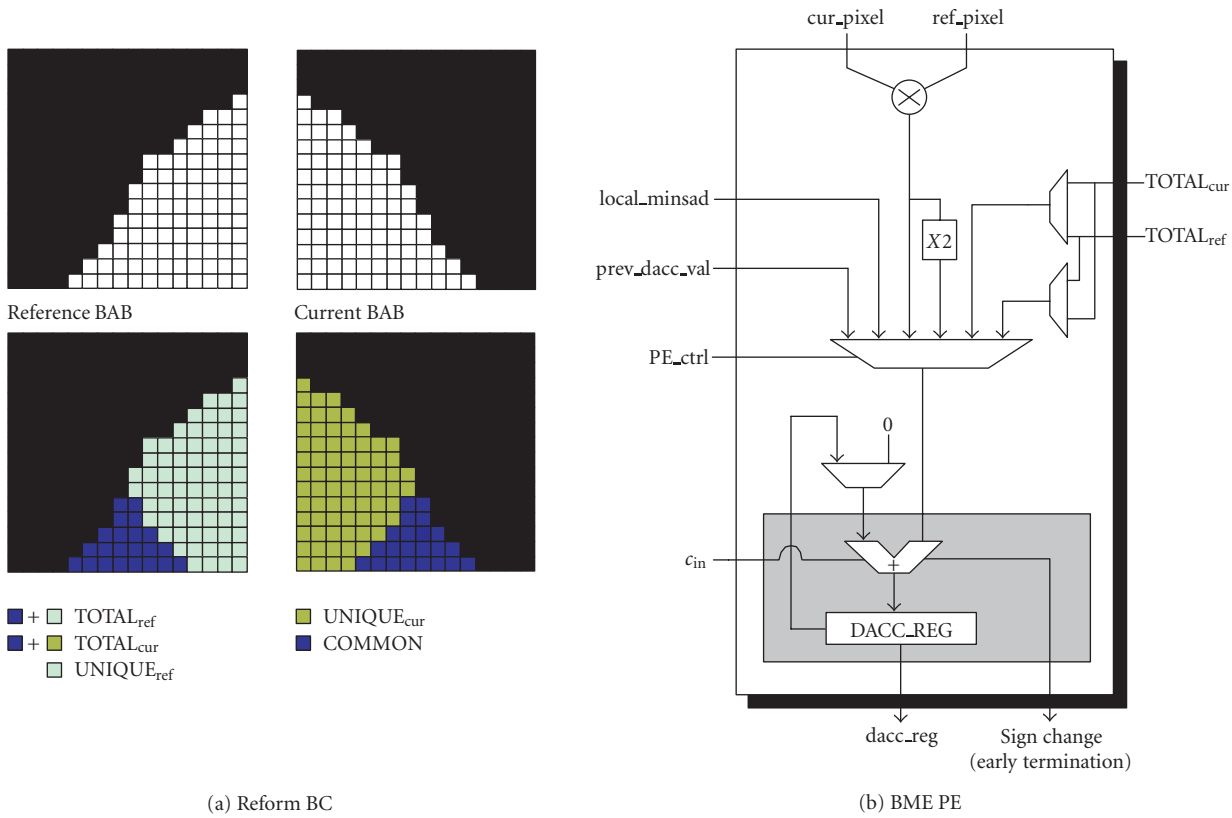
Using these identifies, it follows that

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

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



(a) Reform BC (b) BME PE

Figure 6: Bit count reformulation and BME PE.

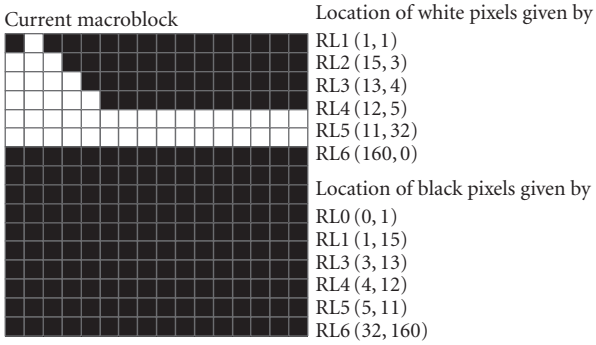


Figure 7: Regular and inverse RLC pixel addressing.

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 TOTAL$_{cur}$ is greater than TOTAL$_{ref}$, (12) is used instead of (11). Since run length coding the reference BAB is not feasible, UNIQUE$_{ref}$ 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,

$$\text{SAD} = \text{TOTAL}_{\text{cur}} - \text{TOTAL}_{\text{ref}} + 2 \times \text{UNIQUE}_{\text{ref}}. \quad (12)$$

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 DACC_REG. During the next cycle TOTAL$_{cur}$/TOTAL$_{ref}$ is added to DACC_REG (depending if TOTAL$_{ref}$[MSB] is 0 or 1, respectively, or if TOTAL$_{ref}$ is larger than TOTAL$_{cur}$). On the next clock cycle, DACC_REG is de-accumulated by TOTAL$_{ref}$/TOTAL$_{cur}$ again depending on whether TOTAL$_{ref}$[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

Table 3: BME synthesis results and benchmarking.

Architecture	Tech (μm)	Cycle count			Gates	PGCC	f (MHz)	Power (mW)	P' (mW)	E' (nJ)	EDP′ (fJs)
		Max	Min	Average							
Natarajan et al. [25]	n/a	1039	1039	1039	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Lee et al. [23]	n/a	1056	1056	1056	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Chang et al. [18]	0.35	1039	1039	1039	9666	1.00×10^7	40	n/a	n/a	n/a	n/a
Proposed	0.09	65 535	2112	6554	10 117	6.63×10^7	100	1.22	1.22	80	5240

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, $\text{TOTAL}_{\text{ref}}$ can be updated in one clock cycle. This is done by subtracting the previous row or column (depending on search window movement) from $\text{TOTAL}_{\text{ref}}$ 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 repartition 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 $f_{\max}$ 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 (6.63×10^7). 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.

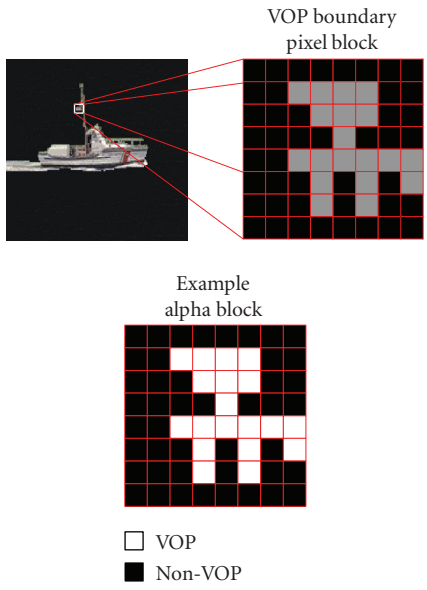


Figure 8: Example VOP boundary block.

4.2. Prior art review

Le and Glesner have proposed two SA-DCT architectures—a recursive 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 8×8 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_buff_A_r and N_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_buff_i_r] in the next clock cycle (where $i \in \{A, B\}$ depends on the interleaved selection signal). The 4-bit register N_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

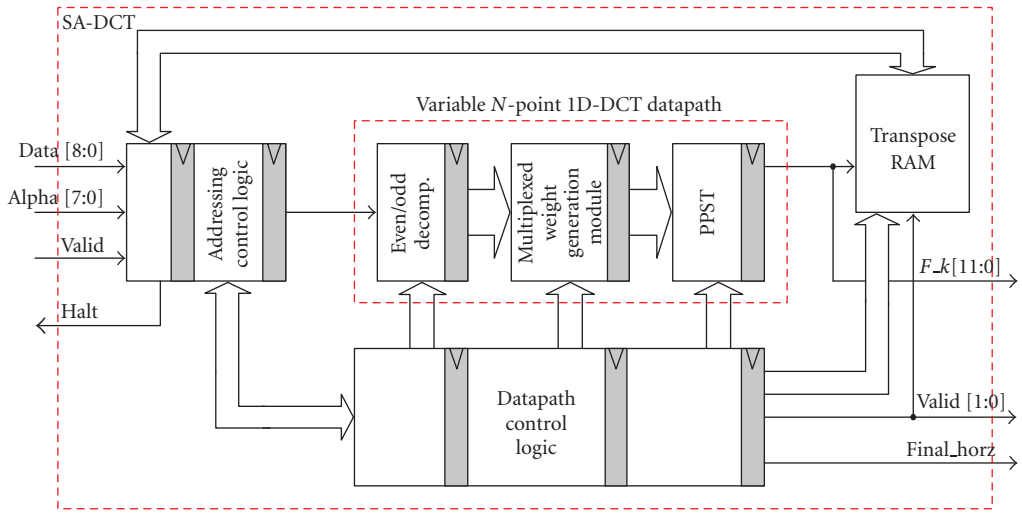


Figure 9: Top-level SA-DCT architecture.

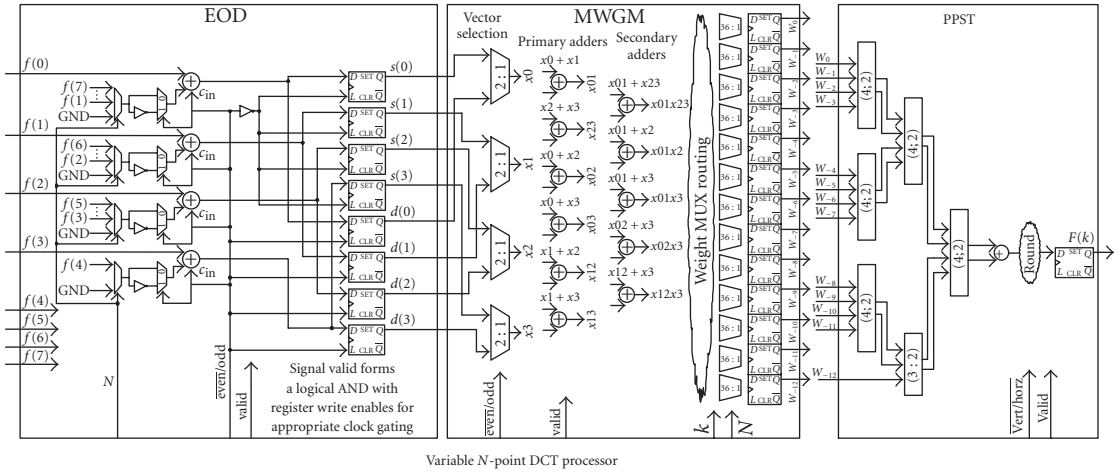


Figure 10: Variable N-point 1D DCT processor.

to minimise the number of accesses. The same scheme ensures horizontal packing is done without redundant shift cycles.

The TRAM is a 64-word $\times$ 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 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

Table 4: SA-DCT synthesis results and benchmarking.

Architecture		Tech (μm)	Cycle count Max	Cycle count Min	+	$*$	Gates	PGCC	f (MHz)	Power (mW)	P' (mW)	E' (nJ)	EDP′ (fJs)
Le and Glesner [29]	◊	0.7	10	3	11	5	n/a	n/a	40	n/a	n/a	n/a	n/a
Tseng et al. [30]	★	0.35	n/a	n/a	n/a	n/a	n/a	n/a	66	180.00	4.33	n/a	n/a
Chen et al. [32]	◊	0.35	124	14	1	0	3157	3.9×10^5	83	n/a	n/a	n/a	n/a
	★		1984	14	n/a	n/a	5775	3.9×10^7		12.44	0.55	12.58	289
Lee et al. [33]	◊	0.35	11	3	11	4	17 341	1.9×10^5	66.7	n/a	n/a	n/a	n/a
Proposed approach	◊	0.09	10	3	27	0	12 016	1.2×10^5	11	n/a	n/a	n/a	n/a
	★		142	79	31	0	25 583	3.6×10^6		0.36	0.36	6.18	60

Key: ◊ ⇒ datapath only, ★ ⇒ datapath and memory.

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 $\overline{\text{even}}$/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 μm TCBN90LP technology yields a gate count of 25 583 and a maximum theoretical operating frequency $f_{\max}$ 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 multiplier 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 VO 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 opaque 8×8 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 8×8 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 8×8 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 8×8 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 autoalignment 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 8×8 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 8×8 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-

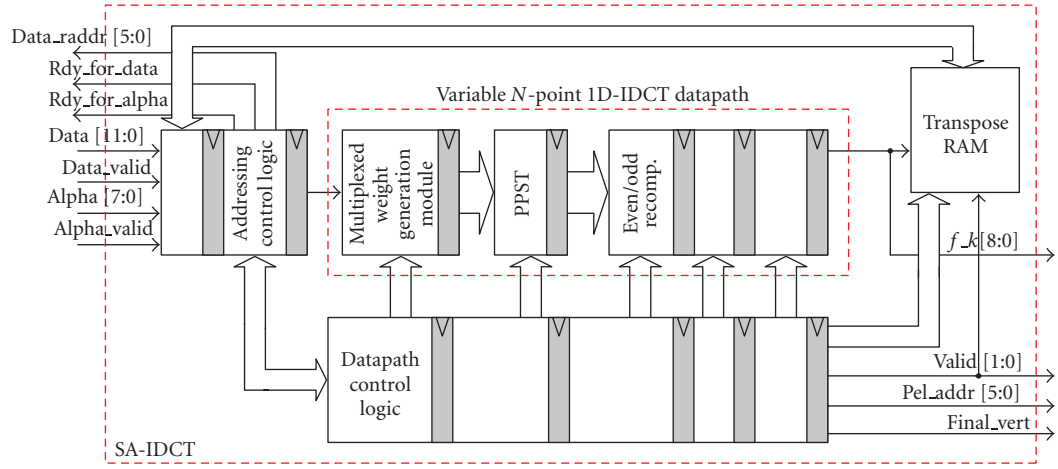


Figure 11: Top-level SA-IDCT architecture.

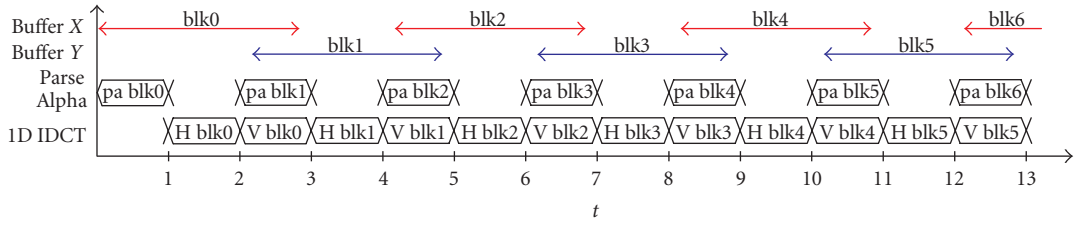


Figure 12: SA-IDCT ACL data processing pipeline.

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 16 N 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 8×8 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

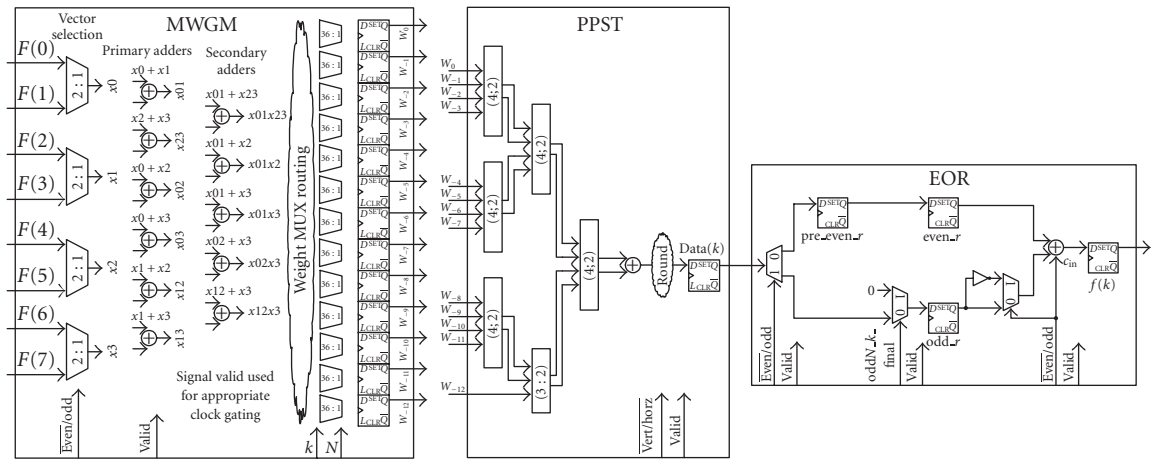


Figure 13: Variable N-point 1D IDCT processor.

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 $8 \times N\_\mathrm{curr}[k] + k$. Then $N\_\mathrm{curr}[k]$ 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 beside 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., $f[N-1], f[0], f[N-2], f[1], \dots$). 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, $(f(0), f(7), f(1), f(6), \dots)$ for $N = 8$. This data ordering eliminates the need for data buffering that is required if the sequence generated is $(f(0), f(1), f(2), f(3), \dots)$. 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 μm TCBN90LP technology yields a gate count of 27518 and a maximum theoretical operating frequency $f_{\max}$ 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 (5.2×10^6 versus 3.9×10^7). 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

Table 5: SA-IDCT synthesis results and benchmarking.

Architecture		Tech (μm)	Cycle count Max	Cycle count Min	+	*	Gates	PGCC	f (MHz)	Power (mW)	P' (mW)	E' (nJ)	EDP′ (fJs)
Chen et al. [32]	◊	0.35	124□	14	1	0	3157	3.9×10^5	83	n/a	n/a	n/a	n/a
	★		1984□	14	n/a	n/a	5775	3.9×10^7		12.44	0.55	13.11	313
Hsu et al. [37]	◊	0.18	8□	3	12	4	n/a	n/a	62.5	n/a	n/a	n/a	n/a
	★		84□	n/a	n/a	n/a	377 685	3.2×10^7		467.04	77.84	104.62	141
			10■	n/a				3.8×10^6		55.60	9.27	1.48	0.2
Proposed approach	◊	0.09	12□	5	24	0	9780	1.2×10^5	14	n/a	n/a	n/a	n/a
	★		188□	125	32	0	27 518	5.2×10^6		0.46	0.46	6.18	8

Key: ◊ ⇒ datapath only, ★ ⇒ datapath and memory.
Operating modes: □ ⇒ no zero skipping, ■ ⇒ zero skipping implemented.

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 (5.2×10^6 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/lw_pwr_arch_des_meth_Landman/Landman94.pdf.

[8] G. K. Yeap, *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 '02)*, 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 J. 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.-C. 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] D. 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 transform," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 8, 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.