Open access • Journal Article • DOI:10.1109/TCSVT.2008.2009250

Multimode Embedded Compression Codec Engine for Power-Aware Video Coding System — Source link

Chih-Chi Cheng, Po-Chih Tseng, Liang-Gee Chen

Institutions: National Taiwan University, Novatek

Published on: 01 Feb 2009 - IEEE Transactions on Circuits and Systems for Video Technology (IEEE)

Topics: Codec, Adaptive Multi-Rate audio codec, Variable-Rate Multimode Wideband, Data compression and Lossy compression

Multi-Mode Embedded Compression Codec Engine for Power-Aware Video Coding System

Chih-Chi Cheng, Po-Chih Tseng, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*— In a typical portable multi-media system, external access, which is usually dominated by block-based video content, induces more than half of total system power. Embedded compression (EC) effectively reduces external access caused by video content by reducing the data size. In this paper, an algorithm and a hardware architecture of a new type EC codec engine with multiple modes are presented. Lossless mode, and lossy modes with rate control modes and quality control modes are all supported by single algorithm. The proposed four-tree pipelining scheme can reduce 83% latency and 67% buffer size between transform and entropy coding. The proposed EC codec engine can save 62%, 66%, and 77% external access at lossless mode, half-size mode, and quarter-size mode and can be used in various system power conditions. With TSMC 0.18 μm 1P6M CMOS logic process, the proposed EC codec engine can encode or decode CIF 30 frame per second video data and achieve power saving of more than 109mW. The EC codec engine itself consumes only 2mW power.**

I. Introduction

As the evolution of manufacturing process in VLSI, silicon area has become less and less important. On the contrary, power plays a more and more important role in VLSI design and will be a critical design issue at least through 2009 [1], [2]. As suggested in [3], the power of DRAM I/O occupies more than 60% system power in a typical multi-chip portable multimedia system. The block-based video content dominates the external access of a video encoding system due to the huge amount of data. Moreover, in a decoding system, power induced by accessing the system bus and display buffer when displaying is also important in total system power [4].

Bus encoding and embedded compression are most commonly-used methods to reduce the bus access power. A bus encoding system reduces the bus access power by reducing the signal transition on the bus. Additional buses are required for bus encoding techniques to indicate the signal change such that the system bus structure has to be altered [5] [6] [7]. The popular bus protocols such as AMBA [8] thus can not be used. Moreover, the bus encoding technique will increase the data size, and the off-chip memory access will also be increased.

This work was supported in part by National Science Council, Republic of China, under the grant number 95-2752-E-002-008-PAE.

Chih-Chi Cheng and Liang-Gee Chen are with DSP/IC Design Lab, Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan, R.O.C. (e-mail: {ccc, lgchen}@video.ee.ntu.edu.tw).

Po-Chih Tseng is with NovaTek Microelectronics Corporation, Hsinchu, Taiwan, R.O.C. (e-mail: borchih@video.ee.ntu.edu.tw).

Digital Object Identifier: xx.xxxx/TCSVT.2008.xxxxxx

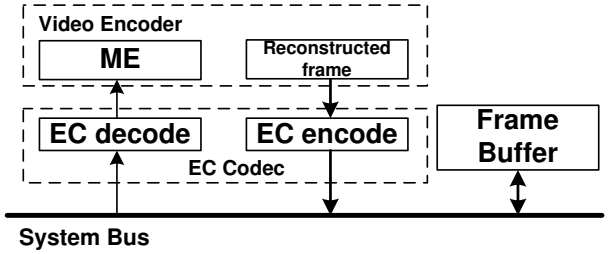

Fig. 1. The scheme for a video encoder with EC engine to reduce external access when ME and MC.

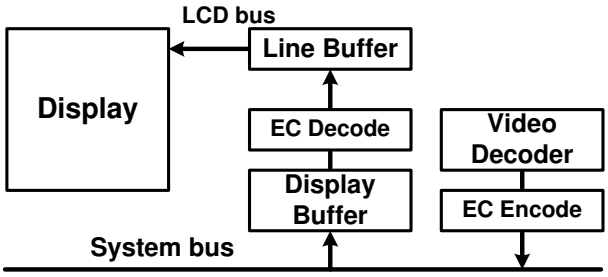

Fig. 2. The scheme for decoding system with EC engine to reduce the system bus and display buffer access when displaying.

An embedded compression (EC) engine reduces the bus access power of video data by compressing the data directly. Block-based video data generated by video coding systems are compressed macroblock by macroblock before they are sent to system bus and hence reduce not only the bus-access power but also the off-chip memory access. Figure 1 shows the scheme for a video encoder with block-based embedded compression engine to reduce the external access. During Motion Compensation (MC), the pixels in motion-compensated frame have to be written into frame buffer. The EC encoder compresses those motion-compensated pixels to reduce the bus and DRAM access. During Motion Estimation (ME), the pixels in searching region of previous encoded frame have to be read from the frame buffer. The EC decoder de-compresses the compressed pixels in previous frame, and passes the de-compressed pixels to the video encoder. Figure 2 shows the presented scheme for a decoding system with EC engine to reduce the system bus and display buffer access when displaying in a similar way. On average, the external access can be reduced to the reciprocal of the compression-ratio (CR) in the EC engine. For example, if CR is equal to two, then the external access can be halved.

In recent years, many algorithms and implementations of embedded compression engines for various applications are presented [4], [9]–[15]. Existing EC algorithms can be divided

into two groups: lossy EC algorithms and lossless EC algorithms.

There are unique advantages in both lossless EC algorithms and lossy EC algorithms. Lossy embedded compression with fixed compression ratio [4], [9]–[13], [15] can guarantee the size of compressed data of each macroblock (MB). Thus it is able to reduce not only external access but also the required external memory size. However, lossy EC is not suitable for applications in close-loop video coding system if the high video quality is necessary. It will cause drifting effects or so called catastrophic error propagation, and severe quality drop will be induced with a large GOP size [9], [10]. Therefore, lossy EC is more suitable for being used in the low-power modes of a power-aware system where quality is of much less importance or in an open-loop coding scheme without drifting effect. Such open-loop schemes include bi-directional predicted frames, motion-compensated temporal filtering scheme, and Scalable Video Coding (SVC) systems with or hierarchical-B scheme. On the contrary, lossless EC can guarantee no quality loss of video data, and hence no drifting effect exists in close-loop video coding systems with uncertain compressed MB size [14]. Lossless EC cannot reduce the memory size until the EC engine maintains a table indicating the bitstream size of each macroblock. However, the lossless EC can still effectively reduce the access power of external memory and system bus with guaranteed video quality.

From the discussion above, a configurable EC codec with both lossless and lossy embedded compression modes can provide a good trade-off between power consumption and visual quality. Therefore, a multi-mode EC codec can be used in a variety of quality/power requirements and perfectly suitable for a power-aware video system.

In this paper, the algorithm, VLSI architecture, and chip implementation for a multi-mode embedded compression codec engine is proposed for power-aware systems. This work is characterized as follows: 1) This is a multi-mode embedded compression codec supporting lossless compression and lossy compression with bitstream size modes and quality control modes. With the adopted Set-Partitioning In Hierarchical Trees (SPIHT) algorithm, all functionalities can be supported by a unified algorithm. 2) A hardware-oriented scheduling of SPIHT, four-tree pipelining scheme, is proposed. The proposed four-tree pipelining scheme can reduce 83% latency, 67% buffer size between DWT and SPIHT, and at least 55% chip area compared to direct implementation of the SPIHT algorithm. 3) VLSI architecture of the EC codec engine is also proposed. The state buffer in SPIHT algorithm is eliminated. Most parts of hardware circuits of EC encoding and EC decoding are shared in the proposed hardware architecture.

This paper is structured as follows. Section II introduces the SPIHT algorithm adopted in proposed EC engine, and the proposed four-tree pipelining scheme will be presented in section III. The corresponding VLSI architecture will be presented in section IV, and the experimental results will be shown and discussed in section V. Finally, section VI will give a conclusion to the proposed multi-mode EC codec engine.

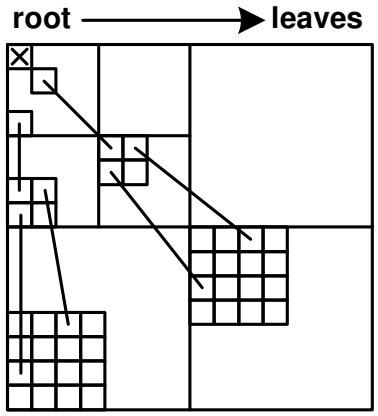

Fig. 3. Illustration of hierarchical tree in SPIHT algorithm.

II. The Adopted SPIHT Algorithm

A. Overview of SPIHT Algorithm

There are two major issues in the selection of coding algorithm for a multi-mode embedded compression codec. The first one is the functionality. To support the power-aware functionality of video coding systems, the embedded compression algorithm must support both lossless encoding and lossy encoding. Therefore, those algorithms support only lossless encoding such as CALIC [16] are not considered. Furthermore, for lossy coding, the encoded bitstream size must not exceed the allocated storage size in off-chip memory. The encoding algorithms that cannot guarantee the encoded bitstream size with one-pass encoding are also not considered [17] [18].

The second issue is the trade-off between the algorithm complexity and coding efficiency. Embedded compression is to reduce the off-chip bus and off-chip memory access power by reducing the data size. The coding efficiency of adopted compression algorithm therefore directly links to the achieved power reduction. JPEG 2000 [19] can achieve very good coding efficiency with higher complexity. The reported JPEG 2000 codec chips all have die sizes above $20mm^2$ with $0.18\mu m$ technology [20] [21] [22]. If JPEG 2000 algorithm is adopted for embedded compression, only the tile buffer between DWT and EBCOT will require more than 30000 logic gates. This gate count is more than the gate count of whole EC codec in our chip implementation. The algorithm in JPEG is simple. However, according to [23], the encoded bitstream size of JPEG is $1.8\times$ of bitstream size of JPEG 2000.

SPIHT algorithm is an image coding algorithm with embedded-coding property presented by Said and Pearlman [24]. According to table II of [25], the quality difference between JPEG 2000 (JP2K) and SPIHT is about 0.15dB, and SPIHT outperform JPEG 2000 in lossless cases. SPIHT supports both lossless and lossy encoding, and it generates bitstream with embedded coding properties. Therefore, the bitrate control is very simple, and the required functionalities can be fully supported as will be discussed in Sec. II-B.

In the SPIHT algorithm, one DWT-transformed image is partitioned into many hierarchical trees. As shown in Fig. 3, a hierarchical tree comprises either a single LL-band coefficient

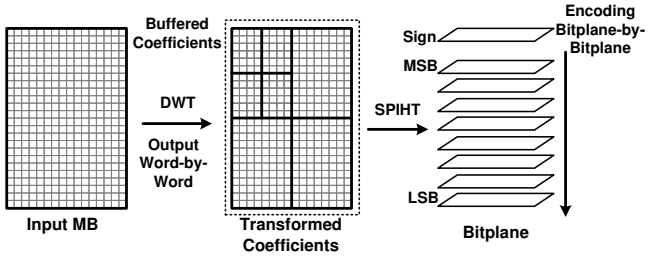

Fig. 4. Illustrations of the data flow of SPIHT encoding and the dataflow mismatch between word by word dataflow of DWT and bitplane by bitplane dataflow of SPIHT engine.

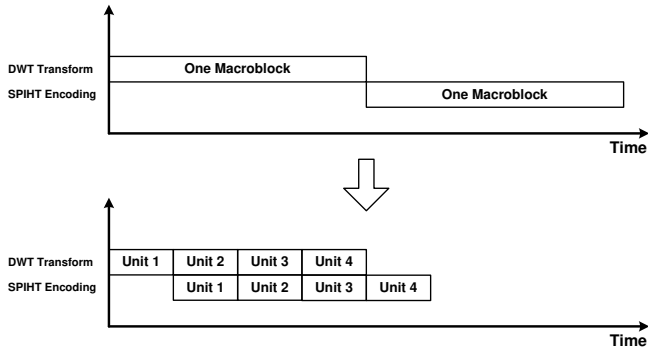

Fig. 5. An illustration of the change of encoding flow by dividing one macroblock into four coding units.

or one LL-band coefficient and low-level coefficients from HL-band, LH-band, or HH-band. The coding efficiency of SPIHT algorithm mainly comes from exploiting the inter-level redundancy by use of this hierarchical tree structure. After partitioning the DWT-decomposed image into hierarchical trees, SPIHT algorithm processes the hierarchical trees bitplane by bitplane, and from the roots to leaves at each bitplane. The encoding flow of SPIHT algorithm is shown in Fig. 4. According to the data in [24], the quality difference between SPIHT algorithm with arithmetic coding and without entropy coding is about 0.3dB in most bitrates. Aiming for a low-cost implementation, this work focuses on SPIHT implementation without entropy coding.

B. Achieving Functionalities of Multi-Mode EC Engine

There are many good inherent properties that make SPIHT coding algorithm suitable for implementation of multi-mode EC codec engine. Firstly, lossless and lossy embedded compression modes are both supported since SPIHT algorithm supports both lossless and lossy coding. Secondly, the fixed compression-ratio (CR) modes can be achieved by simply terminating the encoding/decoding process when the desired size is reached. This is because SPIHT is an embedded coding algorithm. Thirdly, simple quality control can be achieved by truncating bitplanes when encoding, since SPIHT is a bitplane coding algorithm. The distortion induced by truncating the bitplanes can also be estimated well [26]. Finally, the SPIHT algorithm is one of the most simplest coding algorithms with the above-mentioned properties and comparable coding efficiency. This is beneficial for low power VLSI implementation.

C. Design Challenges of Implementing SPIHT in EC Engine

Although there are many good properties that make SPIHT algorithm suitable for EC engine, there are still two main disadvantages for SPIHT algorithm in VLSI design.

The first disadvantage of SPIHT algorithm is the large buffer size between DWT and SPIHT. As can be seen in Fig. 4, DWT transforms the input image pixel-by-pixel, while the SPIHT engine encodes the DWT coefficients bitplane-by-bitplane. Moreover, being different from the moving-window access in embedded block coding of JPEG 2000 [19], SPIHT requires image-level access. This means that when coding one bitplane, the entire current bitplane must be available. This mismatch of the dataflow between DWT and SPIHT coding engine makes it necessary to have a buffer storing DWT coefficients of entire image. This also contributes a long latency since SPIHT module can start encoding only after the finish of DWT transform.

The second disadvantage of SPIHT algorithm is the large buffer inside SPIHT module. There are three buffers defined in the SPIHT algorithm: List of Insignificant Sets (LIS), List of Insignificant Pixels (LIP), and List of Significant Pixels (LSP). The contents of buffer are pixel coordinates. Therefore, the bitwidth is $2\log_2 L$ (L is the image width). In the worst case, the LIS can have $L^2/4$ pixels, and LIP and LSP can have L^2 pixels. Therefore, the required buffer size can be $4.5\log_2 L$ bits. This buffer size is even larger than the image itself.

Several wavelet-based image coding algorithms have been proposed. To reduce the buffer size, no-list SPIHT algorithm is presented [27] with very small quality drop compared to SPIHT algorithm. However, the buffer inside SPIHT engine with size more than a quarter of image size is still required, and this is still too large for a low-cost design. Chrysafis [17] has proposed a line-based coding scheme with very low memory requirement. One advantage of SPIHT over this algorithm is the capability of generating embedded bitstream. In the algorithm of [17], the bitstream size is controlled by adjusting the step of a "dead zone quantizer" before entropy coding, and the encoded bitstream size is therefore unknown before the finish of encoding. In lossy embedded compression applications, however, the encoded bitstream size of each macroblock must not exceed the target bitstream size to avoid the insufficiency of memory space allocation.

III. Proposed Four-Tree Pipelining Scheme

A. Four-Tree Pipelining in SPIHT Engine

As mentioned in Section II-C, the image-level access defined in original SPIHT algorithm induces low hardware utilization, long encoding/decoding latency, and two large-sized buffers. Dividing one macroblock into smaller pieces and encode/decode each piece separately can greatly alleviate all these problems. Firstly, both the two large-sized buffer, DWT coefficient buffer and SPIHT state variable buffer, have a size proportional to the size of encoding unit. Therefore, the size of these two buffers can be greatly reduced by using a smaller coding unit. Secondly, a proper pipeline scheme can be designed and applied between DWT module and SPIHT module such that both DWT hardware and SPIHT hardware

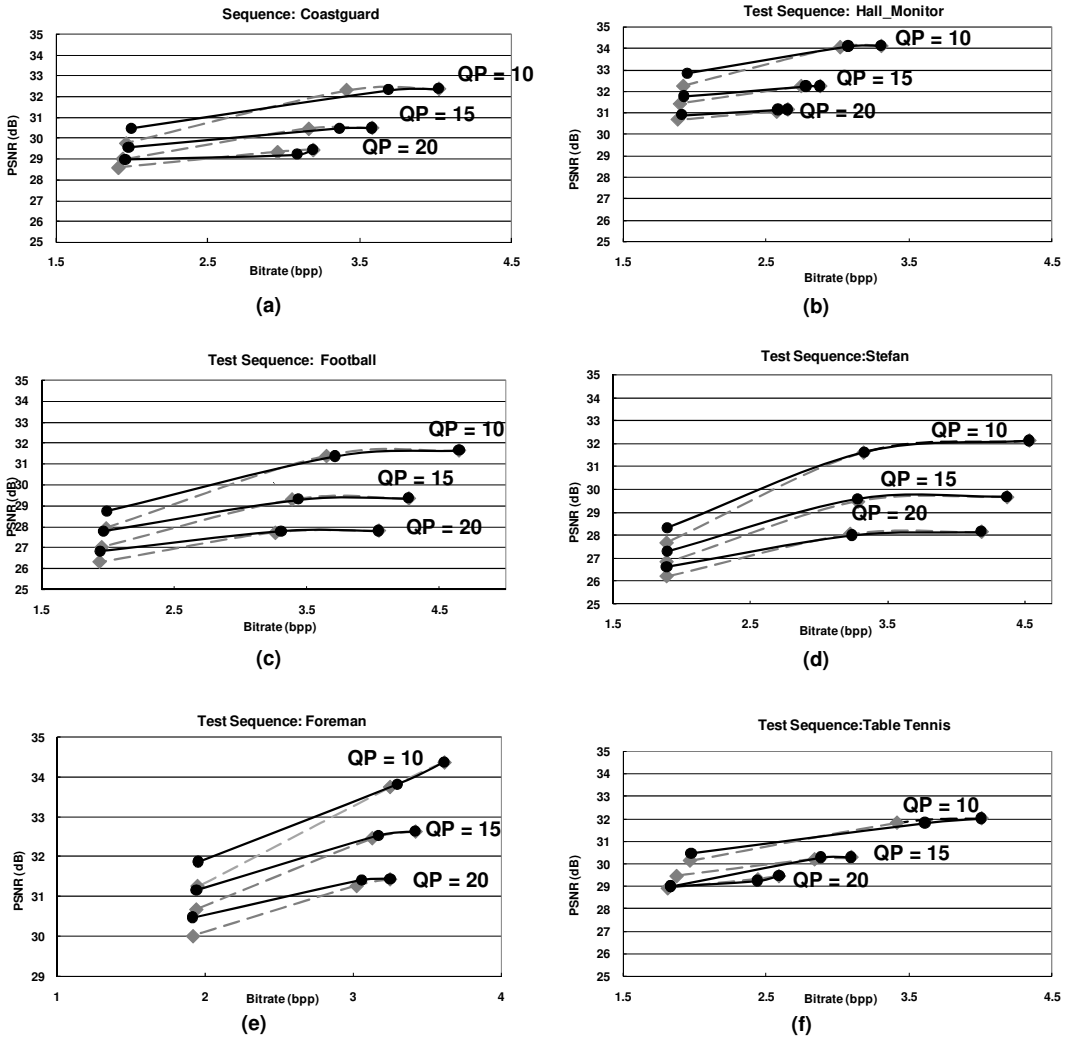

Fig. 7. Comparisons of coding efficiency of SPIHT algorithm (black solid line) and the four-tree pipelined SPIHT (grey dashed line). The test sequences comprise CIF 30 frame-per-second reconstructed videos generated by MPEG-4 simple profile encoder with different quantization parameters (QPs). The results in the figure are the averaged coding efficiency after those sequences are processed by embedded compression macroblock-by-macroblock.

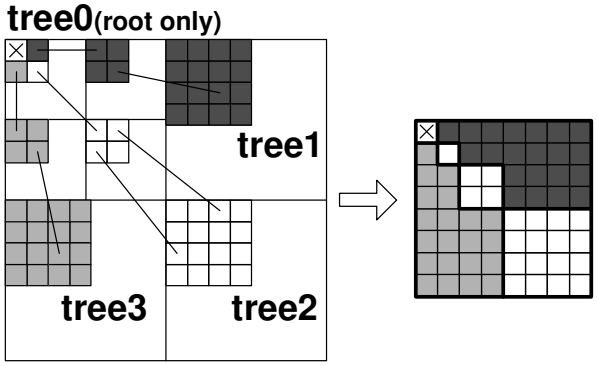

Fig. 6. The proposed algorithm takes four adjacent hierarchical trees (three complete trees and one tree without descendent) in DWT transformed coefficients as a coding unit of SPIHT engine

can work concurrently. Figure 5 shows an illustration of the change of encoding flow of by dividing one macroblock into four coding units. Both the hardware utilization problem and the encoding latency problem can be solved by pipelining in this way.

Using a small coding unit can effectively reduce the buffer size, shorten the latency, and increase the hardware utilization. However, in general, encoding small coding units separately leads to only a local optimization, and the coding efficiency will be degraded. How to choose a coding unit which can maintain the good coding efficiency of SPIHT algorithm is the critical issue to be discussed. The proposed four-tree pipeline scheme is to take adjacent four hierarchical trees as an unit as shown in Fig. 6. As can be seen in Fig. 6, these four trees comprise one LL-band tree with only one coefficient (*tree0*), one HL-band tree (*tree1*), one HH-band tree (*tree2*), and one LH-band tree (*tree3*). These four adjacent hierarchical trees can form a square of $2^{n+1} \times 2^{n+1}$ pixels, where n is the number of decomposition levels, and this square is one coding unit.

There are three main reasons to adopt this four-tree structure. The first reason is that the four-tree structure completely retains the original tree structure in SPIHT, thus the inter-level correlation in a tree is preserved. The inter-level correlation is the main source of the good coding efficiency in SPIHT algorithm. Furthermore, we've found that this change of coding order among trees have no influence on coding efficiency

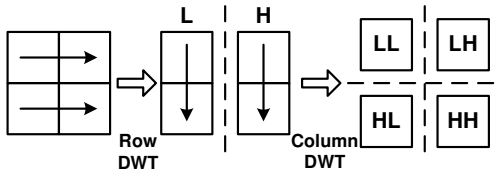

Fig. 8. An example of dataflow in single level 2-D DWT of four pixels

in lossless coding. That is, the proposed four-tree pipelining scheme can achieve exactly the same coding efficiency as that of the original SPIHT algorithm in lossless mode.

The second reason is that the adjacent four trees correspond to the same block of input macroblock. Therefore the most important part of correlation between hierarchical trees is preserved. Figure 7 shows the comparisons of coding efficiency of SPIHT algorithm (solid line) and the four-tree pipelined SPIHT (dashed line). The test sequences comprise CIF 30 frame-per-second reconstructed videos generated by MPEG-4 simple profile encoder with different quantization parameters (QPs). The results in the figure are the averaged coding efficiency after those sequences are processed by embedded compression macroblock-by-macroblock. Three modes are tested: lossless compression mode, half-size mode with target compression ration (CR) to be equal to 2, and quarter-size mode with target CR to be equal to 4. As can be seen from Fig. 7, the proposed four-tree pipeline scheme can have exactly the same coding efficiency as SPIHT algorithm in lossless mode, and the quality drop induced by the four-tree pipeline scheme is very small in other modes.

The final reason is that we've derived a DWT scheme that can match this dataflow to further reduce the buffer size between DWT and SPIHT engine, as will be discussed in the next section.

B. Four-Tree Pipelining in DWT

To reduce the buffer size between DWT and SPIHT engine, the dataflow between them must be matched. During DWT process, coefficients of the four subbands (LL, LH, HL, and HH) are outputted together as shown in Fig. 8. That is, each subband has the same number of outputted coefficients in a period of time. However, in the trees structure shown in Fig. 3, a tree comprises coefficients only from two subbands of different levels (the root is from LL band, and the rest is from LH, HL, or HH band). This leads to a mismatch of dataflow between DWT and SPIHT. The minimum number of trees that matches the dataflow of DWT is four. As shown in Fig. 6, the number of coefficients in each subband at the same level are the same. Therefore, an input macroblock can be divided into blocks with four trees outputted after one block is processed.

C. Overall Scheduling of Four-Tree Pipelining Scheme

As discussed in section III-B, DWT can be performed block by block. Combined with the four-tree pipelining in SPIHT engine shown in section III-A, the whole EC process of a macroblock can be performed block by block. Figure 9 shows the proposed four-tree pipelining scheme of one macroblock. The four-tree pipelining scheme divides the input MB into 6

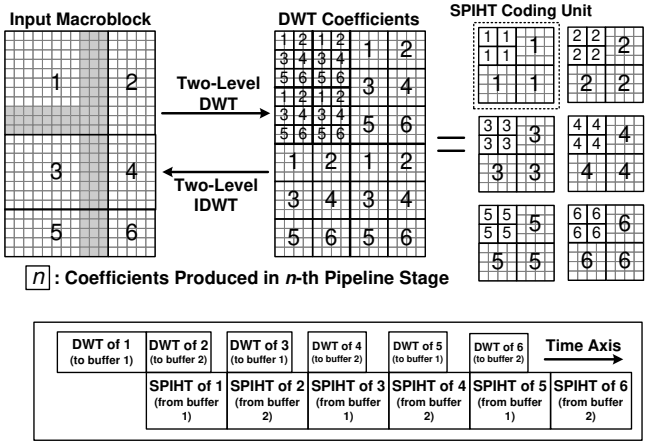

Fig. 9. The proposed four-tree pipelining scheme with encode scheduling

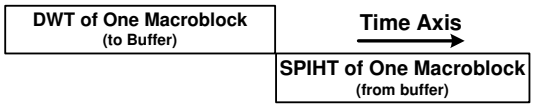

Fig. 10. The encode scheduling of direct implementation of SPIHT algorithm.

smaller blocks as indexed from 1 to 6 in Fig. 9, and each input block is transformed into DWT coefficients corresponding to four hierarchical trees. When encoding, DWT is processing n-th block while SPIHT is processing $(n-1)$-th block in the same pipeline stage as indicated at the bottom of Fig. 9. In general, the size of each input block in Fig. 9 is unequal to be transformed into four hierarchical trees because DWT has latency cycles (the gray region) in most filters such as $(5,3)$ or $(9,7)$ filter commonly used in JPEG 2000. With other simpler DWT transform like S-transform [28], the input block size can be equal. In our final implementation, the S-transform is adopted for the best coding efficiency in virtually all near-lossless and lossless cases. The proposed four-tree pipelining scheme, however, can be adopted with all DWT filters.

Form the discussion above, the buffer size between DWT and SPIHT can be reduced to two-sets of four hierarchical trees rather than the whole macroblock. Compared with the scheduling of direct implementation shown in Fig. 10, the latency of the EC engine is reduced to the duration of performing DWT on a set of four hierarchical trees rather than the duration of performing DWT on the whole macroblock.

IV. Proposed VLSI Hardware Architecture for Multi-Mode EC Codec Engine

A. Utilized Algorithms for EC Codec Engine

The proposed EC codec engine adopts the SPIHT algorithm discussed in section II to support lossless mode, fixed compression-ratio mode, and quality control modes. Aiming for a low cost VLSI implementation, the proposed EC codec engine is scheduled according to the proposed four-tree pipelining scheme presented in section III.

Because the coding unit of the presented EC engine is only one macroblock with luminance and chrominance components (Y:16×16 pixels, U:8×8 pixels, V:8×8 pixels), there will be no increase in coding efficiency with decomposition level

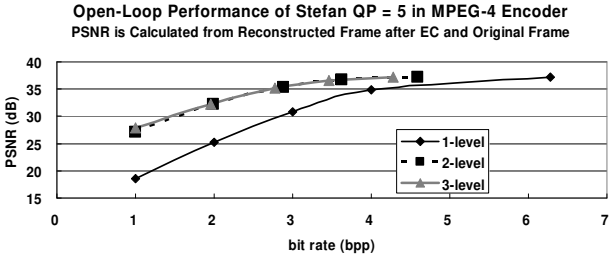

Fig. 11. A typical example of coding efficiency between different decomposition levels of DWT

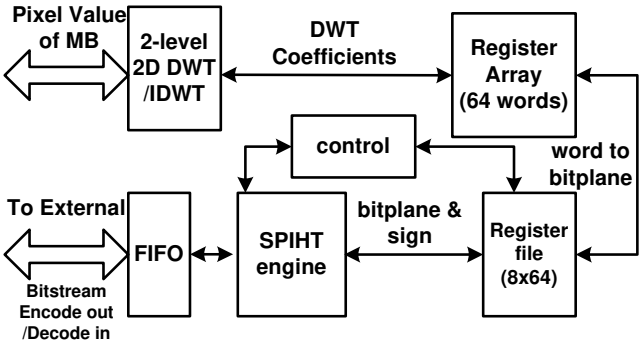

Fig. 12. The proposed system architecture for multi-mode EC codec engine

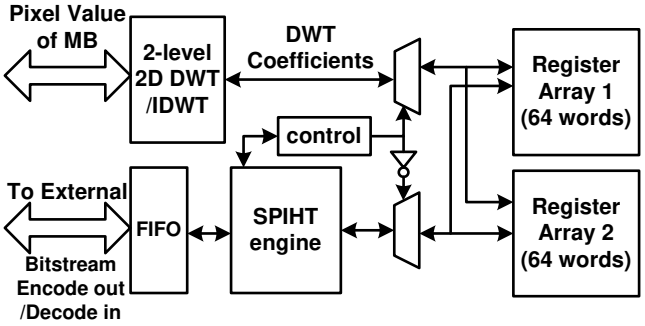

Fig. 13. Alternative system architecture avoiding bubble cycles

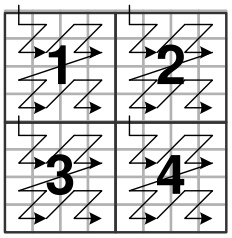

Fig. 14. The sequence of encoding/decoding one coding pass in one bitplane. The number indicating the clock cycle corresponding region is processed, and the arrows indicating the sequence of coding within one clock cycle.

of DWT more than two levels. Figure 11 shows a typical example of coding efficiency between different decomposition levels of DWT. Therefore, the proposed SPIHT architecture is designed for two-level DWT. With two-level DWT, the size of four hierarchical trees is 64 coefficients, which is the size of processed data in one four-tree-pipeline stage. With two-level DWT in four-tree pipelining scheme, 67% buffer size between DWT and SPIHT and 83% latency can be reduced, respectively.

B. Proposed System Architecture

Figure 12 shows the proposed system architecture of the multi-mode embedded compression codec engine. The arrows are all bidirectional because of the difference in dataflow between encoding and decoding. The encoding flow is from the pixel value of a macroblock through DWT and SPIHT to the external, and decoding flow is exactly in the reverse direction. The following discussion will be presented mainly from the encoding perspective and the decoding is just its reverse process.

The input pixel value of a macroblock is firstly passed into DWT module. During two-level DWT, coefficients of four hierarchical trees are outputted to a register array coefficient-by-coefficient. After DWT, there are eight bubble cycles loading the decomposed coefficients into an 8×64 register-file (the behavior is just the same as SRAM) bitplane-by-bitplane. A register file has better storage density and requires less area compared with registers. If the bubble cycles are not desired, an alternative system architecture, as shown in Fig. 13, can be adopted. In the alternative system architecture, two register array buffers are used in a ping-pong mode.

The register file therefore has to be read only once for every bitplane to move bitplane data into 64 bits temporary registers in SPIHT engine, and it can be turned off at other time. This is also another reason utilizing register file for implementation.

After SPIHT coding, the encoded bitstream is outputted through a FIFO. The FIFO comprises a bitstream buffer and control circuits. The FIFO will output after the amount of data accumulated in bitstream buffer exceeds the bitwidth of external bus.

C. Proposed Two-Level SPIHT Codec Engine Architecture

In the special case of two-level DWT utilized, SPIHT algorithm can be further simplified. In a general SPIHT algorithm, three lists are maintained: List of Insignificant Pixel (LIP), List of Insignificant Sets (LIS), and List of Significant Pixels (LSP). Three lists totally require 9.9 KBytes without four-tree pipelining and 1.2 KBytes if the four-tree pipelining scheme is adopted. However, in the special case of two-level DWT utilized, the three lists can be eliminated by exploiting the equivalent possible encoding/decoding signal paths in the SPIHT algorithm.

The coding process of each bitplane is divided into three passes in hardware implementation: LIP pass, LIS pass, and LSP pass. Each pass encodes/decodes the data in the corresponding buffer that originally exists in SPIHT algorithm. Whether one coefficient belongs to the current coding pass or not can be judged by simple combinational circuits with exploiting properties of the SPIHT algorithm and two-level DWT.

The coding process of each pass in one bitplane is planned to be accomplished in four cycles, with sixteen coefficients processed per cycle. Figure 14 shows the sequence of encoding/decoding one coding pass in one bitplane. The coding sequence in one clock cycle is also shown. In general, SPIHT algorithm cannot be implemented with fixed order of coding.

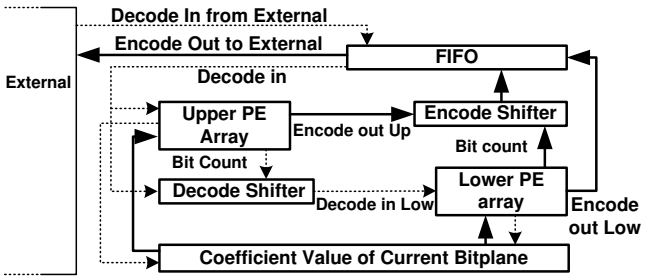

Fig. 15. The architecture of SPIHT codec engine. Solid lines indicate the dataflow of encoding, and the dashed lines are decoding dataflow.

However, in the special case of two-level coefficients, the proposed fixed coding order can make a coding path equivalent to SPIHT algorithm.

Sixteen processing elements (PEs) connected in order shown in Fig. 14 are utilized to encode/decode coefficients within one clock cycle. They are further divided into two groups, eight PEs each.

When encoding, the input of one PE is the bitstream which contains the encoded results of previous PEs. The job of one PE is to calculate the bit count that it will output and to shift the input right by this bit count. After the shifting process, the PE appends its own encoded bits at the leftest side of the shifted input bitstream and passes the appended bitstream to the next PE.

When decoding, the input of one PE is the remaining bitstream where the bits decoded by previous PEs have been removed. The job of one PE is to calculate the bits that it will decode in exactly the same way as in encoding. After that, it removes the bits decoded by it from rightest side of input and shifts the input bitstream right by the bit count before passing the shifted bitstream to the next PE.

Please note that the direction of bitstream in one PE is reversed for the encoding and decoding processes. A PE appends the bits encoded by itself at the leftest side of its input when encoding. However, a PE removes the bits decoded by itself at the rightest side of its input when decoding. This leads to the same shifting process in one PE for the encoding and decoding processes. The calculation of bit count and the encoded/decoded bits are also the same between encoding and decoding. This makes the PE hardware highly reused between encoding process and decoding process.

Figure 15 shows the proposed VLSI architecture for SPIHT codec engine. The solid lines indicate the encoding dataflow, and the dashed lines are the decoding dataflow. The sixteen PEs are divided into two PE arrays with eight PEs in each array to reduce the bitstream length which a PE will shift. This will induce shifters to combine or distribute the bitstream. When encoding, the *Upper PE Array* and *Lower PE Array* generate their own bitstreams and bit counts, the bits generated by these two PE arrays will be combined and sent to FIFO by *Encode Shifter*. When decoding, *Decode Shifter* will shift the input by the number of bits that the *Upper PE Array* has decoded and send the remaining part to the *Lower PE Array*. Please note that only the shifters in the proposed architecture are not shared by encoding process and decoding process.

TABLE II

COMPARISONS OF THE QUALITY DROP INDUCED BY EMBEDDED COMPRESSION AT HALF-SIZE MODE.

	[9]	[10]	[12]	[15]	This Work
avg. PSNR drop @ CR=2	0.65dB	0.2dB	0.5dB[1]	0.15dB[2]	0.1dB

[1]This number is the average number in a 12-frame GOP
[2]The encoding unit conprises 9 macroblocks
[3]Except this work, all works support only fixed-bitrate compression

D. Hardware Scheme for Multi-level DWT/IDWT

The proposed four-tree pipelining is independent of the adopted DWT filter type. As discussed in section III-B, DWT can be performed on a macroblock block by block, where processing a block outputs four hierarchical trees shown in Fig. 6. The proposed architecture is thus suitable for all multi-level block-based DWT architectures with output block size 8×8 [29].

V. EXPERIMENTAL RESULTS

Table I shows the open-loop coding efficiency of the proposed four-tree pipelining scheme with S-transform adopted as DWT filter. The test sequences are the reconstructed frame after motion compensation of MPEG-4 simple-profile encoder with Quantization Parameter (QP) of 10, 15, and 20. QP of 10, 15, and 20 correspond to the high quality, medium quality, and low quality sequences.

On average, 62.4% external access can be saved in lossless mode. The lossless mode in the proposed EC engine can be used under the normal operation of a power-aware system, and no quality will be sacrificed. The size of encoded macroblock is not guaranteed, so the external memory size for one macroblock has to be unchanged.

65.3% external access can be saved in half-size mode, and compression ratio of all macroblock is at least two. The average quality drop is 0.1dB. This mode can be used in the low-power modes of power-aware encoding/decoding systems where moderate quality loss is allowed. This mode further guarantee at least 50% size reduction of each macroblock. Therefore, the frame buffer size can also be halved.

On average, 77% external access can be eliminated in quarter-size mode, the compression ratio of all macroblock is at least four. However, there is more quality drop with this mode, and therefore quarter-size mode is more suitable for very low power mode of power-aware encoding/decoding system. In such cases the accomplishment of encoding/decoding process is the most important thing, this quarter-size mode will be very useful since it can reduce 77% external access power. Is can also be used for low-quality video with tolerable video quality drop (0.78dB when QP = 20).

Table III shows the comparisons between the proposed architecture with four-tree pipelining scheme and direct implementation of SPIHT algorithm. Utilizing the proposed four-tree pipelining scheme results in the reduction of latency and buffer size between DWT and SPIHT. Exploiting SPIHT algorithm with two-level DWT eliminates the lists in SPIHT algorithm.

TABLE I

THE OPEN-LOOP CODING EFFICIENCY OF THE PROPOSED EC CODEC WITH S-TRANSFORM AS DWT FILTER. SEQUENCES ARE RECONSTRUCTED FRAMES OF MPEG-4 ENCODER WITH DIFFERENT QP (QUANTIZATION PARAMETER) VALUES.

		Coding Efficiency (Reduced Data Ratio / Quality Loss Induced by Lossy EC)		
QP	Sequence	Lossless Mode	Half-Size (CR = 2)	Quarter-Size (CR = 4)
10	akiyo	69.8% / 0dB	71.1% / 0.40dB	77.8% / 3.25dB
	coastguard	49.2% / 0dB	57.3% / 0.07dB	75.4% / 2.61dB
	foreman	54.3% / 0dB	59.4% / 0.63dB	75.6% / 3.12dB
	mother-daughter	67.8% / 0dB	68.8% / 0.00dB	77.3% / 1.38dB
	silent	58.7% / 0dB	62.3% / 0.05dB	76.0% / 1.85dB
	table-tennis	49.9% / 0dB	57.3% / 0.19dB	75.4% / 1.85dB
	Average	**58.2% / 0dB**	**62.4% / 0.22dB**	**76.2% / 2.34dB**
15	akiyo	71.3% / 0dB	72.4% / 0.01dB	78.1% / 1.50dB
	coastguard	54.6% / 0dB	60.4% / 0.01dB	75.7% / 1.49dB
	foreman	56.7% / 0dB	60.9% / 0.17dB	75.7% / 1.94dB
	mother-daughter	70.9% / 0dB	71.5% / 0.00dB	78.0% / 0.69dB
	silent	64.0% / 0dB	65.7% / 0.01dB	76.3% / 0.8dB
	table-tennis	61.3% / 0dB	64.5% / 0.04dB	76.5% / 0.8dB
	Average	**63.1% / 0dB**	**65.9% / 0.04dB**	**76.7% / 1.2dB**
20	akiyo	72.0% / 0dB	72.8% / 0.00dB	78.3% / 0.98dB
	coastguard	59.6% / 0dB	63.0% / 0.01dB	76.0% / 0.83dB
	foreman	58.8% / 0dB	62.3% / 0.17dB	76.0% / 1.42dB
	mother-daughter	71.1% / 0dB	71.5% / 0.00dB	77.9% / 0.48dB
	silent	66.8% / 0dB	67.8% / 0.06dB	76.5% / 0.46dB
	table-tennis	67.6% / 0dB	69.4% / 0.14dB	77.3% / 0.55dB
	Average	**65.9% / 0dB**	**67.8% / 0.06dB**	**77.0% / 0.78dB**
Overall Average		**62.4% / 0dB**	**65.3% / 0.10dB**	**76.6% / 1.44dB**

TABLE III

THE COMPARISONS BETWEEN THE PROPOSED ARCHITECTURE WITH FOUR-TREE PIPELINING SCHEME AND DIRECT IMPLEMENTATION OF SPIHT ALGORITHM

	Direct Implementation	Proposed EC Codec	Reduction
Latency	240 cycles	40 cycles	83%
Buffer Between DWT and SPIHT	3.84 KB	1.28 KB	67%
State Variable Buffer in SPIHT Engine	9.88 KB	12 bits	99%

Table II compares this work with previous embedded compression works. For hardware area or silicon implementation results are not available, only the coding efficiency is compared here. The quality drop data are the PSNR difference when video processed by video coding systems with and without embedded compression. Average quality drop data with embedded compression running on half-size mode are used in the table.

Figure 16 shows the prototyping chip of the proposed multi-mode EC engine, and Table IV shows the measurement results. The power consumption is 2mW while supporting CIF 30 frame per second encoding/decoding. The encoding/decoding time is 84us for each macroblock. An encoded macroblock will be required by the video coding system after at least 30ms, so the encoding/decoding delay introduced by EC will not cause problems. Without the proposed four-tree pipelining scheme, only the buffer between DWT and SPIHT will occupy 38,400 logic gates. Therefore, the proposed four-tree pipelining reduces at least 55% total gate-count. Moreover, the target timing specification can not be met with the same hardware architecture because the latency is six-time longer.

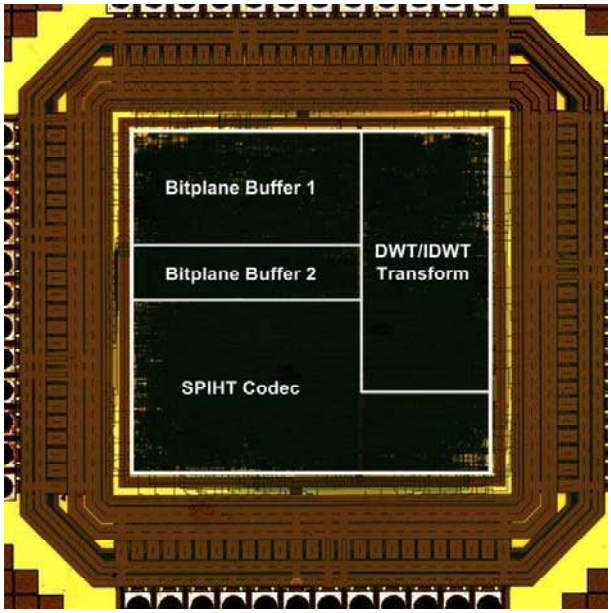

Fig. 16. The die photo of the implemented prototyping chip.

According to the data provided in [3], the external access in a MPEG-4 encoding system supporting QCIF resolution consumes 651mW with 0.25um process. Assuming equal

TABLE IV
THE MEASURED RESULTS OF THE PROTOTYPING CHIP OF PROPOSED MULTI-MODE EC ENGINE. TSMC 0.18 μM 1P6M CMOS PROCESS IS UTILIZED.

Supply Voltage	1.3V
Working Frequency	10MHz
Throughput	CIF (352x288) 4:2:0 @30fps
Lossless EC	Supported
Fixed Compression-Ratio Mode	Half-Size, Quarter-Size
Quality Control	8 Quality Levels
Logic Gate Count	26,932
Buffer Size	8x64 single-port SRAM
Core Size	0.75x0.75 mm^2
Measured Power Consumption	encoding: 1.77mW
	decoding: 2.01mW

oxide thickness, we scale this power consumption into 0.18um technology:

$$\begin{aligned} P_{180} &= P_{250} \times \frac{C_{180}}{C250} \times (\frac{V_{DD_{180}}}{V_{DD_{250}}})^2 \\ &= P_{250} \times (\frac{0.18}{0.25})^2 \times (\frac{1.8}{2.5})^2 \\ &= P_{250} \times 0.268 \\ &= 174.9mW \end{aligned} \quad (1)$$

According the average data reduction ratio listed in Table I, the proposed EC engine can save 109mW, 114mW, and 136mW in 0.18um process with lossless mode, half-size mode, and quarter-size mode, respectively. Because the implemented EC codec can process video with CIF resolution that is 4× as the QCIF resolution in [3], the power saving introduced by this EC codec can be actually even larger. Compared with the 2mW power consumption of EC engine while processing video with CIF format, the power saving is quite huge.

VI. CONCLUSION

In this paper, the algorithm and architecture for a new type embedded compression codec engine with multiple modes are proposed. With the proposed EC codec engine, lossless embedded compression and lossy embedded compression with rate control modes and quality control modes can be all supported by single algorithm based on SPIHT algorithm.

The proposed four-tree pipelining can reduce 83% latency and 67% buffer size between DWT and SPIHT compared with direct implementation of SPIHT algorithm. Proposed hardware architecture of multi-mode embedded compression codec engine shares most hardware circuits between EC encoder and decoder by designing of dataflow. Moreover, 9.9KBytes state buffer in SPIHT algorithm is eliminated by exploiting the properties of two-level SPIHT algorithm.

The implemented EC codec engine can encode or decode at CIF 30fps with 10MHz clock rate and 1.3V power supply. The proposed EC codec engine can save 62%, 66%, and 77% external access with lossless mode, half-size mode, and quarter-size mode and can be used in various scenarios. The gate count is 27K with 8×64 bits buffer and the measured power is 2mW. Compared with the amount the reduced DRAM area and external access power (we estimated as 109mW or more), the area and power overhead is small.

REFERENCES

[1] "International technology roadmap for semiconductors(ITRS) 2003 edition," 2003.

[2] Pat Gelsinger, "Giga-scale integration for tera-ops performance - opportunities and new frontiers," in *Keynote Speech of IEEE Design Automation Conference*, 2004.

[3] T. Nishikawa and et al., "A 60MHz 240mW MPEG-4 video-phone LSI with 16Mb embedded DRAM," in *Digest of Technical Papers, International Solid-State Circuits Conference (ISSCC)*, 2000, pp. 230–231.

[4] Hojun Shim, Naehyuck Chang, and Massoud Pedram, "A compressed frame buffer to reduce display power consumption in mobile systems," in *Proceedings of the Asia and South Pacific Design Automation Conference*, 2004, pp. 819–824.

[5] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, pp. 49–58, Mar. 1995.

[6] M. Olivieri, F. Pappalardo, and G. Visalli, "Bus-switch coding for reducing power dissipation in off-chip buses," *IEEE Transactions on VLSI Systems*, vol. 12, no. 12, pp. 1401–1404, Dec. 2004.

[7] Z. Khan, T. Arslan, and T. Erdogan A, "A novel bus encoding scheme from energy and crosstalk efficiency perspective for AMBA based generic SoC systems," in *IEEE International Conference on VLSI design*, 2005, pp. 751–756.

[8] *AMBA Specification*, http://www.amba.com.

[9] Ulug Bayazit, Lenny Chen, and Robert Rozploch, "A novel memory compression system for MPEG-2 decoders," in *IEEE International Conference of Consumer Electronics*, 1998, pp. 56–57.

[10] M. v.d. Schaar-Mitrea and Peter H.N. de With, "Near-lossless embedded compression algorithm for cost reduction in DTV receivers," in *IEEE International Conference of Consumer Electronics*, 1999, pp. 112–113.

[11] Shawmin Lei, "A quad-tree embedded compression algorithm for memory-saving DTV decoders," in *IEEE International Conference of Consumer Electronics*, 1999, pp. 120–121.

[12] Egbert G.T. Jaspers and Peter H.N. de With, "Embedded compression for memory resource reduction in MPEG systems," in *IEEE Benelux Signal Processing Symposium*, 2002, pp. S02–1–S02–4.

[13] G. M. Callico, A. Nunez, R. P. Llopis, and R. Sethuraman, "Low-cost and real-time super-resolution over a video encoder IP," in *Proceedings of IEEE Fourth International Symposium on Quality Electronic Design*, 2003, pp. 79–84.

[14] Rashindra Manniesing, Richard Kleihorst, Rene van der Vleuten1, and Emile Hendriks, "Implementation of lossless coding for embedded compression," *IEEE Program for Research on Integrated Systems and Circuits/Workshop on Circuits, Systems and Signal Processing*, 1998.

[15] Peter H.N. de With, Peter H. Frencken, and Mihaela v.d. Schaar-Mitrea, "A MPEG decoder with embedded compression for memory reduction," in *IEEE International Conference of Consumer Electronics*, 1998, pp. 60–61.

[16] Xiaolin Wu and N. Memon, "CALIC-a context based adaptive lossless image codec," in *Proceedings of 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1996, pp. 1890–1893.

[17] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 378–389, Mar. 2000.

[18] *JPEG Image Coding System*, ISO/IEC IS 10918-1 — ITU-T Recommendation T.81, 2000.

[19] *JPEG 2000 Image Coding System*, ISO/IEC FDIS15444-1, 2000.

[20] Yu-Wei Chang, Hung-Chi Fang, Chih-Chi Cheng, Chun-Chia Chen, Chung-Jr Lian, Shao-Yi Chien, and Liang-Gee Chen, "124Ms/s pixel-pipelined motion-JPEG 2000 codec without tile memory," in *Digest of Technical Papers, International Solid-State Circuits Conference (ISSCC)*, 2006, pp. 404–405.

[21] Yu-Wei Chang, Chih-Chi Cheng, Chun-Chia Chen, Hung-Chi Fang, and Liang-Gee Chen, "124 MSamples/s pixel-pipelined motion-JPEG 2000 codec without tile memory," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 4, pp. 398–406, Apr 2007.

[22] H. Yamauchi, K. Mochizuki, K. Taketa, T. Watanabe, T. Mori, Y. Matsuda, Y. Matsushita, A. Kobayashi, and S. Okada, "A 1440×1080 pixels 30frames/s motion-JPEG2000 codec for HD movie transmission," in *Digest of Technical Papers, International Solid-State Circuits Conference (ISSCC)*, 2004, pp. 326–530.

[23] D. Taubman and M. Marcellin, "JPEG2000: Standard for interactive imaging," *Proceedings of IEEE*, vol. 90, no. 8, pp. 1336–1357, Aug. 2002.

[24] Amir Said and William A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243 – 250, June 1996.

[25] W. A. Pearlman, A. Islam, N. Nagaraj, and A. Said, "Efficient, low-complexity image coding with a set-partitioning embedded block coder," *IEEE Transactions on Circuits and Sysyems for Video Technology*, vol. 14, no. 11, pp. 1219–1235, Nov. 2004.

[26] C.-C. Cheng, C.-T. Huang, J.-Y. Chang, and L.-G. Chen, "Line buffer wordlength analysis for line-based 2-D DWT," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2006, vol. 3, pp. 924–927.

[27] F. W. Wheeler and W. A. Pearlman, "SPIHT image compression without lists," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2000, pp. 2047–2050.

[28] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Transactions on Image Processing*, vol. 5, pp. 1303–1310, Sept. 1996.

[29] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "Analysis and VLSI architecture for 1-D and 2-D discrete wavelet transform," vol. 53, no. 4, pp. 1575–1586, 2005.