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

Chih-Chi Cheng , Po-Chih Tseng, Chao-Tsung Huang, and Liang-Gee Chen
DSP/IC Design Lab, Graduate Institute of Electronics Engineering and
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
Email: {ccc,pctseng,cthuang,lgchen}@video.ee.ntu.edu.tw

Abstract— **In a typical multi-chip handheld system for multimedia applications, 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 50%, 61%, and 77% external access at lossless mode, half-size mode, and quarter-size mode and can be used in various system power conditions. With Artisan 0.18 μm cell library, the proposed EC codec engine can encode or decode at VGA@30fps with area and power consumption of 293,605 μm^2 and 3.36 mW.**

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 system with multimedia functionalities. 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 of system bus access by video content when displaying is also important in total system power [4].

A block-based embedded compression (EC) engine can compress block-based video data generated by video coding system macroblock by macroblock before they are sent to system bus and hence reduce the bus-access power. Figure 1 shows the presented scheme for a video encoder with block-based embedded compression engine to reduce the external access when loading search range of motion-estimation(ME) and motion compensating(MC). Figure 2 shows the presented scheme for a decoding system with EC engine to reduce the system bus access when displaying. 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]–[10]. Existing EC algorithms can be divided into two groups: lossy EC algorithms and lossless EC algorithms.

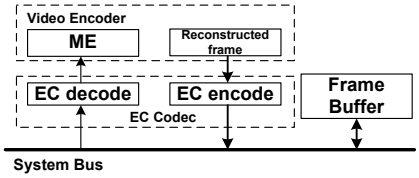


Fig. 1. The presented scheme for a video encoder with EC engine to reduce external access when ME and MC.

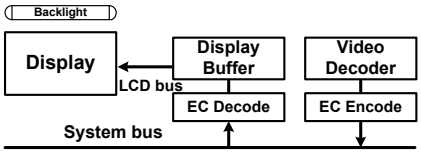


Fig. 2. The presented scheme for decoding system with EC engine to reduce the system bus access when displaying.

Both lossless and lossy EC engines have their own unique advantages. Lossy embedded compression with fixed compression ratio [4]–[9] 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 error propagation and severe quality drop with a large GOP size [5], [6]. Therefore it is more suitable for the low-power modes in a power-aware system where quality is of much less important or an open-loop coding scheme without drifting effect such as B frames or Scalable Video Coding (SVC) systems with Motion-Compensated Temporal Filtering 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 [10]. Lossless EC therefore can not reduce the memory size, but it can still reduce the access power of external memory and system bus.

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

In this paper, the algorithm and VLSI architecture for a multi-mode block-based embedded compression codec engine is presented for power-aware systems. With the adopted Set-Partitioning in Hierarchical Trees (SPIHT) algorithm, both lossless and lossy EC can be supported almost without hardware overhead. Due to the inherent properties of SPIHT,

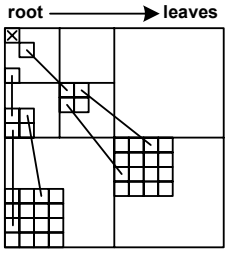


Fig. 3. Illustration of hierarchical tree in SPIHT algorithm.

several quality control modes are also available. A hardware efficient scheduling of SPIHT, four-tree pipelining scheme, is also 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 SPIHT algorithm. High performance VLSI architecture of the EC codec engine is also proposed, and 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. With the proposed multi-mode EC engine, all of lossless mode and lossy modes with rate control or quality control are supported, and the VLSI architecture is of high throughput and area efficient.

This paper is structured as follows. Section II introduces the SPIHT algorithm adopted in proposed EC engine, and the proposed hardware-efficient four-tree pipelining scheme will be presented in section III. The corresponding VLSI hardware algorithm and 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.

II. The Adopted SPIHT Algorithm

A. Overview of SPIHT Algorithm

Image coding using SPIHT is a fully embedded coding algorithm presented by Said and Pearlman in 1996 [11]. Recent experimental results show that SPIHT algorithm maintains good coding efficiency with a simple algorithm [12].

The SPIHT algorithm partitions one DWT-transformed image into many hierarchical trees to exploit the inter-layer redundancy, as shown in Fig. 3. 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.

B. Achieving Functionalities of Multi-Mode EC Engine

The SPIHT coding algorithm has many good inherent properties that make it 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

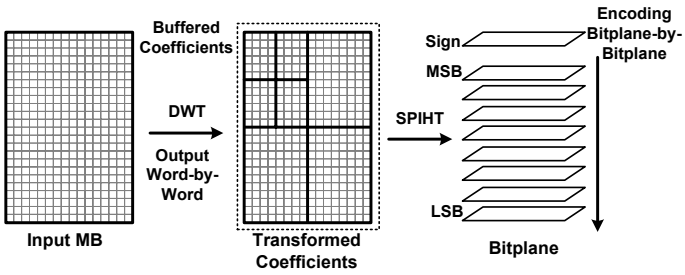


Fig. 4. Illustrations of the data flow of SPIHT encoding and the dataflow mismatch between word by word dataflow of DWT and bitplane by bitplane dataflow of SPIHT engine.

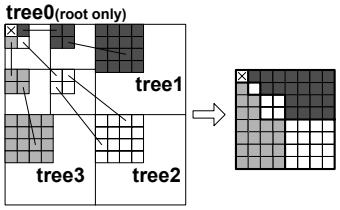


Fig. 5. The proposed algorithm takes four adjacent hierarchical trees (three complete trees and one tree without descendent) in DWT transformed coefficients as a coding unit of SPIHT engine

coding algorithm. The distortion induced by truncating the bitplanes can be estimated. Finally, SPIHT algorithm is one of the most simplest coding algorithms with the above-mentioned properties and comparable coding efficiency. This is necessary for low power VLSI implementation.

C. Design Challenges of Implementing SPIHT in EC Engine

Although the SPIHT algorithm has many good properties that make it suitable for EC engine, there are two main disadvantages for SPIHT 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 is a word-level arithmetic process, while the SPIHT engine encodes/decodes coefficients bitplane by bitplane. Moreover, being different with EBCOT in JPEG 2000, SPIHT requires image-level access. This means that when coding one bitplane, the entire current bitplane must be available. This mismatch of the data flow between DWT and SPIHT coding engine makes it necessary to buffer DWT coefficients of entire image. This also makes the latency between the first input and the first output to be at least the duration of performing DWT of entire image.

The second disadvantage of SPIHT algorithm is the large buffer inside SPIHT engine. In a straightforward implementation of SPIHT, $6L^2 \times \log_2 L$ (L is the image width) bits are needed for the LIS, LIP, and LSP buffers in SPIHT algorithm. This buffer size is even larger than the image itself. To reduce the buffer size, no-list SPIHT is presented [13] with a small quality drop compared to SPIHT. However, the buffer inside SPIHT engine with size more than a quarter of image size is still needed, and this is also too large for a low-cost design.

III. Proposed Four-Tree Pipelining Scheme

A. Four-Tree Pipelining in SPIHT Engine

Because the SPIHT algorithm requires image-level access, the state/coefficient buffer size in SPIHT engine is proportional

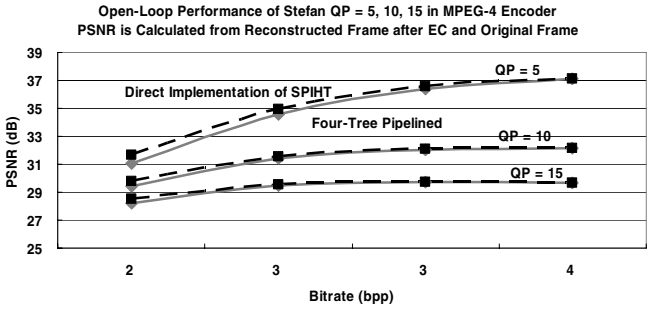


Fig. 6. Comparisons of coding efficiency of SPIHT algorithm with and without four-tree pipelining.

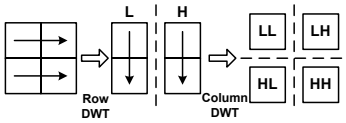


Fig. 7. An example of dataflow in single level 2-D DWT of four pixels

to the image size. Reducing the coding unit size of SPIHT engine can effectively reduce the corresponding state buffer size of SPIHT. However, coding only a subset of image once will lead to only a local optimization, not a global optimization of the original SPIHT algorithm. How to choose a coding unit which can maintain the good coding efficiency of SPIHT algorithm is the issue to be discussed. The proposed four-tree pipelining scheme is to take adjacent four hierarchical trees as a unit, as shown in Fig. 5. There are three main reasons to propose this four-tree structure.

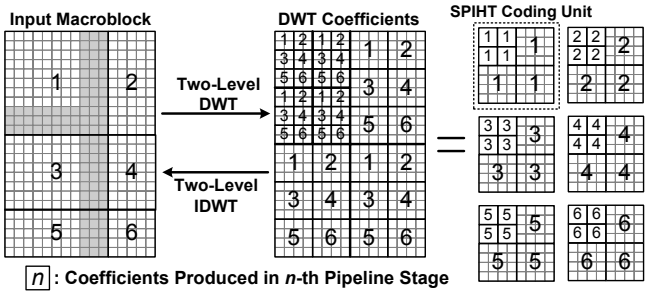
The first one is that the four-tree structure retains the original tree structure in SPIHT, thus the inter-level correlation in a tree is preserved. The inter-level correlation is the main source of the good coding efficiency in SPIHT algorithm. Furthermore, the changes of coding order among trees have no influence on coding efficiency in lossless coding. That is, the proposed four-tree pipelining in SPIHT engine can achieve exactly the same bit-rate in lossless mode if arithmetic coding is not used.

The second reason is that the adjacent four trees correspond to the same block of input image. Therefore the most important part of correlation between hierarchical trees is preserved. Figure 6 shows the comparisons of coding efficiency of SPIHT algorithm with and without four-tree pipelining

The final reason is that there are some architectures of DWT that can match this data flow 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 data flow 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. 7. 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 has coefficients only from two subbands of different levels. This leads to a mismatched data flow between DWT and SPIHT. The minimum number of trees that matches the data flow of DWT is four. As shown in Fig. 5, the number of



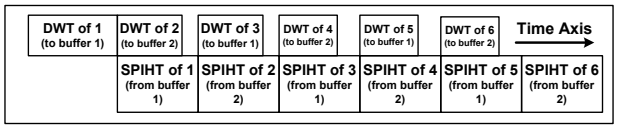


Fig. 8. The proposed four-tree pipelining scheme with encode scheduling

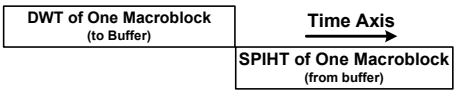


Fig. 9. The encode scheduling of direct implementation of SPIHT algorithm.

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 can be performed on a macroblock block by block. Figure 8 shows the proposed four-tree pipelining scheme of one macroblock. The four-tree pipelining scheme divides the input MB into blocks, and each block outputs four hierarchical trees after DWT. When encoding, DWT is processing n-th block while SPIHT is processing $(n-1)$-th block in the same pipeline stage. The size of each input block in Fig. 8 is unequal because DWT has latency cycles (the gray region) in most filters such as (5,3) filter. With other simpler DWT transform like S-transform [14], the input block size can be equal.

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. 9, the latency of the EC engine is reduced to the duration of performing DWT of four hierarchical trees rather than the duration of performing DWT of 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 color components (Y:16×16, U:8×8,

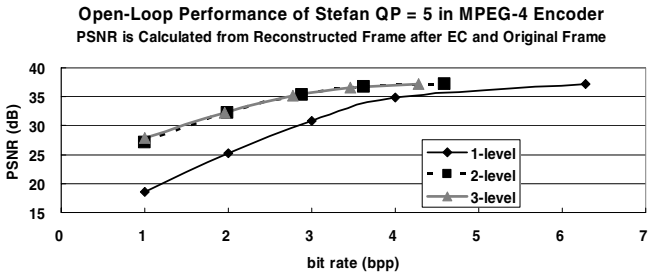


Fig. 10. A typical example of coding efficiency between different decomposition levels of DWT

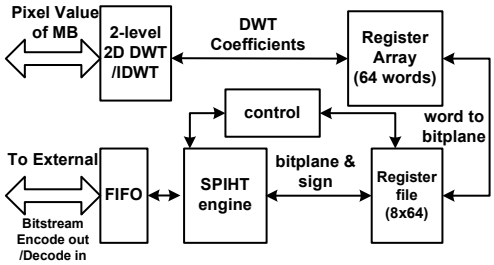


Fig. 11. The proposed system architecture for multi-mode EC codec engine

V:8×8), there will be no increase in coding efficiency with decomposition level of DWT more than two levels. Figure 10 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 data in one 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 11 shows the proposed system architecture of the multi-mode embedded compression codec engine. The arrows are all bidirectional because the difference of data flow 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 since decoding is 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 registers word by word. After DWT, there are eight bubble cycles loading the decomposed coefficients into a 8×64 register-file 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. 12, can be adopted. In the alternative system architecture, two buffers composed of registers is 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 consists of a bitstream buffer, and it

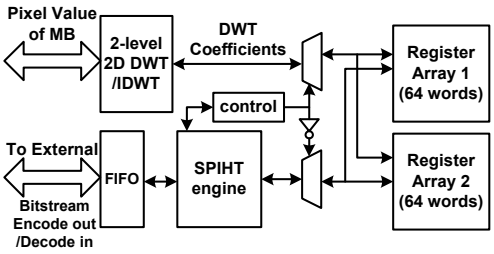


Fig. 12. Alternative system architecture avoiding bubble cycles

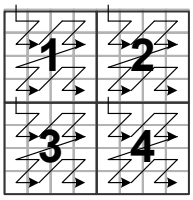


Fig. 13. The sequence of encoding/decoding one coding pass in one bitplane. The number indicating the clock cycle corresponding region is processed, and the arrows indicating the sequence of coding within one clock cycle.

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 with the four-tree pipelining scheme. However, in the special case of two-level DWT utilized, the three lists can be eliminated by exploiting the equivalent possible 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 can be judged from simple combinational circuits by 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 13 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. 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 as the order shown in Fig. 13 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 that, 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.

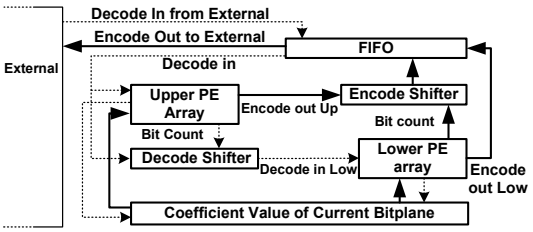


Fig. 14. The proposed VLSI architecture of SPIHT codec engine. Solid lines indicate the dataflow of encoding, and the dotted lines are decoding dataflow.

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.

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 a highly reused PE in SPIHT codec design.

Figure 14 shows the proposed VLSI architecture for SPIHT codec engine. The solid lines indicate the encoding dataflow, and the dotted 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, the *Decode Shifter* will shift the input by the number of bits *Upper PE Array* is decoded and sent the remaining part to the *Lower PE Array*. Note that only the shifters in the proposed architecture are not shared by encoding process and decoding process.

D. Hardware Scheme for Multi-level DWT/IDWT

The proposed four-tree pipelining is independent of the adopted type of DWT filter. 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. 5. The proposed architecture is thus suitable for all multi-level block-based DWT architectures with output block size 8×8 [15].

V. Experimental Results

Table I shows the open-loop coding efficiency of the proposed four-tree pipelining scheme with DWT in (5,3) filter and S-transform.

50% 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.

About 60% external access can be saved in half-size mode, and compression ratio of all macroblock is at least two. Therefore, the frame-buffer size can be halved. The average quality drop is 0.3dB in video sequence with encoding system using QP = 15. This mode can be used in the low-power modes of power-aware encoding/decoding systems for a low-bitrate application where moderate quality loss is allowed. It is also suitable for an open-loop encoding/decoding system such as SVC encoder/decoder targeting for low-cost applications.

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 requirement, this quarter-size mode will be very useful since it can reduce 77% external access power.

Table II 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 the latency of EC codec engine and the buffer size between DWT and SPIHT. Exploiting SPIHT algorithm with two-level DWT eliminates the lists in SPIHT algorithm.

Table III shows the implementation results of the proposed embedded compression codec engine. The power is only 3.36mW with 30MHz clock rate and 1.2V power supply. Without the proposed four-tree pipelining scheme, the buffer between DWT and SPIHT will alone occupy area of 384,000 μm^2. Therefore, the proposed four-tree pipelining reduces at least 55% total chip area. Moreover, the target timing speci-

TABLE II

The Comparisons between the Proposed Architecture with Four-Tree Pipelining Scheme and Direct Implementation of SPIHT Algorithm

	Direct Implement	Proposed
Latency	240 cycles	40 cycles
Buffer Between DWT and SPIHT	3.84 KB	1.28 KB
State Buffer in SPIHT Engine	9.88 KB	12 bits

TABLE III

The Implementation Results of Proposed Multi-Mode EC Engine. Artisan 0.18 μm Cell Library is Utilized.

Supply Voltage	1.2 V
Working Frequency	30 MHz
Throughput (Encoding or decoding)	VGA(640x480) (4:2:0)@30fps
Lossless EC	Supported
Fixed CR mode	CR = 2, CR = 4
Quality Layers	Enable 0~7 Bitplane Trancation
Synthesized Gate Count	26,932
Buffer Size	8x64 bits(One Single-Port Register File)
Synthesized Area	293,605 um^2
Estimated Power	3.36 mW

TABLE I

THE OPEN-LOOP CODING EFFICIENCY OF PROPOSED FOUR-TREE PIPELINING SCHEME WITH (5,3) FILTER AND S-TRANSFORM AS DWT FILTER. SEQUENCES ARE RECONSTRUCTED FRAMES OF MPEG-4 ENCODER WITH DIFFERENT QP VALUES.

	Average Performance (Data Size Reduction Ratio / Quality Drop)					
	Lossless Mode		Half-Size (CR = 2)		Quarter-Size (CR = 4)	
Sequence(QP Value)	(5,3) Filter	S Transform	(5,3) Filter	S Transform	(5,3) Filter	S Transform
foreman(5)	49.6% / 0dB	51.3% / 0dB	55.7% / 0.2dB	58.1% / 0.5dB	75.4% / 3.9dB	76.1% / 5.5dB
stefan(5)	41.3% / 0dB	40.5% / 0dB	55.9% / 0.7dB	57.7% / 1.6dB	75.6% / 6dB	76.6% / 8.4dB
dancer(5)	60.3% / 0dB	65.9% / 0dB	65.9% / 1.6dB	70.9% / 2dB	77.1% / 7dB	79.4% / 6.4dB
mobile(5)	28.3% / 0dB	27.2% / 0dB	52% / 4dB	53.3% / 5dB	75.1% / 9.5dB	75.6% / 10dB
coastguard(5)	43.7% / 0dB	44.8% / 0dB	53.8% / 1.8dB	56% / 2.2dB	75% / 5.2dB	75% / 6.1dB
table(5)	41.5% / 0dB	43.9% / 0dB	53.4% / 3.6dB	56.4% / 2.3dB	75.3% / 5.5dB	76.2% / 5.5dB
Average of QP = 5	**44.1% / 0dB**	**45.6% / 0dB**	**56.1% / 1.9dB**	**58.8% / 2.2dB**	**75.6% / 6.1dB**	**76.5% / 6.9dB**
foreman(10)	53.6% / 0dB	56.9% / 0dB	58.2% / 0.1dB	62.2% / 0.1dB	75.6% / 1.8dB	76.8% / 2.8dB
stefan(10)	43.4% / 0dB	43.8% / 0dB	56.9% / 0.1dB	59.4% / 0.4dB	75.8% / 2.7dB	77.2% / 4.4dB
dancer(10)	62% / 0dB	68.6% / 0dB	66.9% / 0.7dB	72.9% / 0.8dB	77.5% / 4.2dB	80.4% / 3.6dB
mobile(10)	30.4% / 0dB	30.8% / 0dB	52.5% / 1.4dB	54.8% / 2.1dB	75.3% / 4.7dB	75.9% / 5.7dB
coastguard(10)	47.5% / 0dB	49.8% / 0dB	56.3% / 0.9dB	59.2% / 0.5dB	75.2% / 1.9dB	75.2% / 2.7dB
table(10)	46% / 0dB	50.5% / 0dB	55.4% / 1.2dB	60% / 0.8dB	75.3% / 2.6dB	76.5% / 2.4dB
Average of QP = 10	**47.2% / 0dB**	**50% / 0dB**	**57.7% / 0.7dB**	**61.4% / 0.8dB**	**75.8% / 2.9dB**	**77% / 3.6dB**
foreman(15)	55% / 0dB	59.8% / 0dB	59.4% / 0dB	64.5% / 0dB	75.9% / 1dB	77.5% / 1.1dB
stefan(15)	44.5% / 0dB	45.8% / 0dB	57% / 0dB	60.0% / 0.1dB	75.9% / 1.4dB	77.3% / 2.7dB
dancer(15)	62.8% / 0dB	70.1% / 0dB	67.4% / 0.5dB	74% / 0.5dB	77.6% / 2.9dB	80.9% / 2.5dB
mobile(15)	31.8% / 0dB	32.9% / 0dB	52.8% / 0.8dB	55.5% / 1dB	75.3% / 2.9dB	76.2% / 3.8dB
coastguard(15)	50.8% / 0dB	55.3% / 0dB	57.5% / 0.4dB	62.5% / 0.2dB	75.3% / 1.2dB	75.3% / 1.5dB
table(15)	53.4% / 0dB	61.9% / 0dB	60.4% / 0.8dB	67.8% / 0.3dB	75.6% / 1.7dB	78.8% / 1.2dB
Average of QP = 15	**49.7% / 0dB**	**54.3% / 0dB**	**59.1% / 0.4dB**	**64% / 0.3dB**	**75.9% / 1.8dB**	**77.6% / 2.1dB**
Total Average	**47% / 0dB**	**50% / 0dB**	**57.6% / 1dB**	**61.4% / 1.1dB**	**75.8% / 3.6dB**	**77% / 4.2dB**

fication can not be met with the same hardware architecture because the latency is six-times longer.

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 VGA 30fps with 30MHz clock rate and 1.2V power supply. The proposed EC codec engine can save 50%, 61%, and 77% external access with lossless mode, half-size mode, and quarter-size mode, and it can be used in various scenarios. The estimated gate count is 27K with 8×64 bits buffer and the estimated power is 3.36 mW. Compared with the amount the reduced DRAM area are and external access power, 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 60 MHz 240 mW MPEG-4 video-phone LSI with 16 Mb embedded DRAM," in *Digest of Technical Papers of IEEE International Solid-State Circuits Conference*, 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] Ulug Bayazit, Lenny Chen, and Robert Rozploch, "A novel memory compression system for MPEG2 decoders," in *IEEE International Conference of Consumer Electronics*, 1998, pp. 56–57.

[6] 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.

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

[8] Egbert G.T. Jaspers and Peter H.N. de With, "Embedded compression for memory resource reduction in MPEG systemss," in *IEEE Benelux Signal Processing Symposium*, 2002.

[9] Gustavo M. Callico, Antonio Nunez, Rafael Peset Llopis, and Ramanathan Sethuraman, "Low-cost and real-time super-resolution over a video encoder ip," *IEEE*, 2003.

[10] Rashindra Manniesing, Richard Kleihorst1, Rene van der Vleuten1, and Emile Hendriks, "Implementation of lossless coding for embedded compression," *IEEE ProRISC*, 1998.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.