

Fine-scalable SPIHT Hardware Design for Frame Memory Compression in Video Codec

Sunwoong Kim¹, Ji Hun Jang², Hyuk-Jae Lee¹, and Chae Eun Rhee²

Abstract—In order to reduce the size of frame memory or bus bandwidth, frame memory compression (FMC) recompresses reconstructed or reference frames of video codecs. This paper proposes a novel FMC design based on discrete wavelet transform (DWT) - set partitioning in hierarchical trees (SPIHT), which supports fine-scalable throughput and is area-efficient. In the proposed design, multi-cores with small block sizes are used in parallel instead of a single core with a large block size. In addition, an appropriate pipelining schedule is proposed. Compared to the previous design, the proposed design achieves the processing speed which is closer to the target system speed, and therefore it is more efficient in hardware utilization. In addition, a scheme in which two passes of SPIHT are merged into one pass called merged refinement pass (MRP) is proposed. As the number of shifters decreases and the bit-width of remained shifters is reduced, the size of SPIHT hardware significantly decreases. The proposed FMC encoder and decoder designs achieve the throughputs of 4,448 and 4,000 Mpixels/s, respectively, and their gate counts are 76.5K and 107.8K. When the proposed design is applied to high efficiency video codec (HEVC), it achieves 1.96% lower average BDBR and 0.05 dB higher average BDPSNR than the previous FMC design.

Index Terms—Embedded compression, image

compression, VLSI implementation, high efficiency video codec (HEVC), parallel architectures

I. INTRODUCTION

Video codecs store reconstructed or reference frames into external frame memory to execute an inter prediction which is based on temporal correlation. Typically, video codecs access the frame data through a bus. The stored data are often recompressed, and thereby reducing the frame memory size or bus bandwidth. This method is called frame memory compression (FMC) [1-10]. FMC reduces frame memory access costs but requires an additional hardware. Furthermore, low read/write latency is demanded not to affect the processing speed of video codecs. Therefore, unlike video codecs such as high efficiency video codec (HEVC), low computation is important rather than high coding efficiency in FMC. Therefore, temporal correlation is not exploited and spatial correlation is used restrictedly.

Recently, the resolution size is exceeding 4K UHD (3840×2160) and video codecs require very high-throughput. The throughput of FMC is increasing as well. Until now, various FMC algorithms have been proposed. Lee *et al.* propose a differential pulse code modulation (DPCM) - variable length coding (VLC)-based FMC algorithm [1]. However, this algorithm generates the bitstream with a variable length. In addition, the bitstream has dependence inside, and therefore the high-throughput is not supported. To solve this problem, FMC algorithms without VLC have been studied. Kim *et al.* propose prediction and coding methods which are effective in hardware implementation [3]. The prediction

Manuscript received Oct. 31, 2016; accepted Mar. 27, 2017

¹Inter-university Semiconductor Research Center, Department of Electrical Engineering, Seoul National University, Seoul 151-742, Korea

²Department of Information and Communication Engineering, Inha University, Incheon 22212, Korea
E-mail : chae.rhee@inha.ac.kr

method called hierarchical average and copy prediction (HACP) offers high prediction accuracy using diverse average values within a block. Moreover, the coding method called significant bit truncation (SBT) enables the number of the bits generated within a block to be calculated in advance, leading to the high-throughput. Guo *et al.* propose a prediction method called multi-mode DPCM and averaging (MDA) which effectively compresses various images. In addition, a coding method called semi-fixed length (SFL) is proposed to predict the bitstream length in advance, thereby achieving very high-throughput [6]. Despite the high-throughput, HACP-SBT and MDA-SFL only support lossless coding, and therefore have limited applications. To support lossy coding, the quantization method in [1] can be applied to those algorithms. However, HACP-SBT and MDA-SFL insert dummy bits to make the bitstream length foreknowable. As the target compression ratio (CR) increases, the ratio of those dummy bits in the bitstream increases, resulting in decrease in coding efficiency.

In case that the CR and image quality are the most important factors to consider when choosing an FMC algorithm, lossless coding is most appropriate. It is difficult to assure that the same FMC is used for the video encoder and decoder. Therefore, the FMC, which is lossy coding, may make a reference frame mismatch between encoding and decoding, which is continuously accumulated. For this reason, HACP-SBT and MDA-SFL, which are used as FMC in video codecs, only consider lossless coding. However, lossless coding algorithms are not suitable for hardware systems. Since the length of the bitstream generated by the lossless coding is variable, the frame memory size and bandwidth vary as well. Furthermore, random access on the reference frame required by video codecs is difficult. For example, since the bitstream having a variable length is sequentially stored in frame memory, it is difficult to calculate an initial memory address for each search range. If the bitstream is discontinuously stored in frame memory to support easy random access, the bus utilization to access the frame memory is significantly reduced. For example, dummy bits are inserted into the data transfer and the burst mode of the bus protocol is not fully utilized. This reduces the compression gain in the bus bandwidth utilization.

Discrete wavelet transform (DWT) - set partitioning in

hierarchical trees (SPIHT) is one of lossy coding algorithms [11]. The DWT-SPIHT achieves an accurate target CR and supports lossless coding as well. In addition, this algorithm is based on bit-plane coding and effective in coding efficiency due to errors occurring in lower bit-planes. However, the processing speed is slow due to the dynamic processing order. In addition, existing dependence inside bitstream and each pass interferes with the speed. To solve this problem, various methods including hardware architectures have been studied [5, 7, 12-15]. Kim *et al.* propose an SPIHT hardware design of which throughput is about 1 bit-plane per cycle [7]. In this design, the processing cycle is fixed regardless of the block size, and therefore the throughput can be continuously improved by increasing the block size. However, this scalable throughput has some problems with hardware implementation. First, a 2D DWT is composed of two 1D DWTs whose transform directions are horizontal and vertical. When the horizontal transform ends, the vertical transform starts. Therefore, as the block size increases, latency and temporal memory cost increase. Second, there is a stage of transposing coefficients to bit-planes between DWT and SPIHT, which is implemented in memory elements. As the block size increases, the size of the memory element proportionally increases. Third, when the height and width of a square block are doubled, the processing speed increases four times. Therefore, achieving the target system speed is difficult and the hardware utilization decreases. Fourth, as the block size increases, the number of bits to be coded in each pass of SPIHT increases with each cycle. Consequently, the burden on the packer module to shift and merge the generated bits increases. Likewise, the complexity of the parser module increases.

This paper proposes a DWT-SPIHT-based FMC design that supports fine-scalable throughput and area-efficiency. Contributions are set apart from previous designs as follows. First, an SPIHT design that finely increases throughput is proposed. To this end, multi-cores with small block sizes are used in parallel instead of the single core with a large block size. An appropriate pipelining schedule is also proposed. Second, the number of the passes in SPIHT is decreased to reduce the hardware overhead that is increased by the multi-cores. Finally, the proposed FMC design is integrated into the

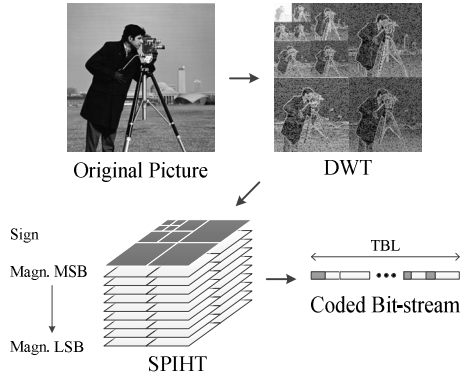


Fig. 1. DWT-SPIHT algorithm.

HEVC encoder system with the AXI bus, and recompresses reconstructed/reference frames. Considering FMC coding efficiency and hardware implementation advantages, it is a reasonable solution to use a lossy coding algorithm in systems where the video codec is connected to external frame memory through a bus. The inconsistency between the encoder and the decoder, which is caused by the lossy coding algorithm, is mitigated by using periodic intra frames.

The rest of this paper is organized as follows. In Section II, the previous DWT-SPIHT algorithm is introduced. In Sections III, a DWT-SPIHT design which shows fine-scalable throughput is proposed. In Section IV, an integration with the AXI-based HEVC encoder hardware is presented. The implementation results and compression performances of the proposed FMC design are presented in Section V, and Section VI concludes the paper.

II. DWT-SPIHT

This section presents the DWT-SPIHT algorithm that is the basis of the proposed algorithm. The DWT-SPIHT algorithm which is one of transform-based coding algorithms encodes input pictures as shown in Fig. 1. First of all, it executes 2D DWT on the input picture. The 2D DWT is implemented by using vertical and horizontal 1D DWTs. Wavelet transformed coefficients are temporarily stored in memory and then transposed into bit-plane units. The transposed bit-planes are transferred to SPIHT from the most significant bit-plane (MSB) to the least significant bit-plane (LSB). The bit-planes have a spatial-orientated tree (SOT) data structure in which coefficients that have the same spatial orientation but

different frequency band levels are connected. It is likely that the higher band level is insignificant if the corresponding lower band level is insignificant. This is called the zero-tree hypothesis. Based on this hypothesis, SPIHT represents a set which does not have any significant bit ‘1’ by symbol ‘0’. This process, called the significance test, is shown in (1).

$$S_n(T) = \begin{cases} 1, & \max_{(i,j) \in T} \{c_{i,j}\} \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In (1), T and $S_n(\cdot)$ represent a set to be tested and the significance test on the n -th bit-plane, respectively. In addition, $c_{i,j}$ represents a coefficient at location (i, j) in T and the threshold is set by 2^n when the n -th bit-plane is coded. If any of coefficients in T is larger than or equal to the threshold, the significance test outputs ‘1’. The tested set is then divided into several sub-sets and those sub-sets are tested again. The coding efficiency of SPIHT is depending on how many sets are represented by the symbol ‘0’. SPIHT terminates when all the input picture data are coded or the bitstream length meets the target bit length (TBL). Therefore, this algorithm can achieve the target CR exactly. Moreover, as coding proceeds from MSB to LSB, errors occur in lower bit-planes. Therefore, SPIHT shows high coding efficiency among various lossy coding algorithms. Despite its many advantages, SPIHT has a problem of slow processing speed. This is because the conventional SPIHT codes data in dynamic order. To solve this problem, SPIHT algorithms which code data in fixed order have been proposed [5, 7, 12-15]. In these algorithms, coding efficiency is slightly reduced but processing speed is significantly improved.

Up to our knowledge, the SPIHT hardware design which shows the highest throughput in both of encoder and decoder is the design in [7]. In this design, SPIHT is composed of three passes: sorting pass (SP), first refinement pass (FRP), and refinement pass (RP). Originally, the parallel processing in SPIHT is restricted because of two dependences: The first exists between SPs of different frequency band levels and the second exists between SP and FRP of the same frequency band level. Kim *et al.* propose a pipelining schedule to avoid those dependences, and therefore all passes are processed in parallel. As a result, the encoder and decoder achieve

high-throughput of about one bit-plane per cycle. Fig. 2 shows the pipelining schedule of the encoder design in [7] of which the decomposition level is two. In this figure, vertical and horizontal axes represent passes in each band and execution time, respectively. In addition, one rectangle and the number inside represent one cycle time and the bit-plane number, respectively. During the first two cycles, represented in white color, all passes do not work in parallel to avoid the dependences. However, all passes work in parallel from the third cycle, which is represented in light gray color. The delayed bit-planes are represented in dark gray color. As this design processes about one bit-plane per cycle, regardless of the block size, throughput can be improved by increasing the block size. However, the required hardware resources, such as buffers, significantly increase as the block size increases. Moreover, hardware utilization may decrease because a fine control of throughput is difficult. Therefore, it is required to divide one large block into several small blocks and process them in parallel to achieve an area-efficient and high-throughput design.

III. THE PROPOSED SCHEMES FOR FINE-SCALABLE SPIHT HARDWARE DESIGN

This section proposes a fine-scalable SPIHT design for area-efficient hardware implementation.

1. Multi-core SPIHT

The parallel and pipelined method in [7] enables the number of processing cycles to be fixed regardless of the block size. Therefore, as the block size increases, the number of generated bits per cycle increases, consequently resulting in high-throughput. However, the hardware utilization may greatly decrease depending on the target speed or the incoming input data speed. Fig. 3 shows an example of execution in [7] in which FMC is connected to a bus with 64 bit-width and the DWT decomposition level is set by two. In this figure, input data come into the FMC with the target processing speed of 8 pixels/cycle in the burst mode. Fig. 3(a) shows the execution of an 8×8 block while Fig. 3(b) shows that on a 16×16 block. Gray and white squares in Fig. 3 represent sign and magnitude bit-planes, respectively, and dotted squares represent the delayed bit-planes

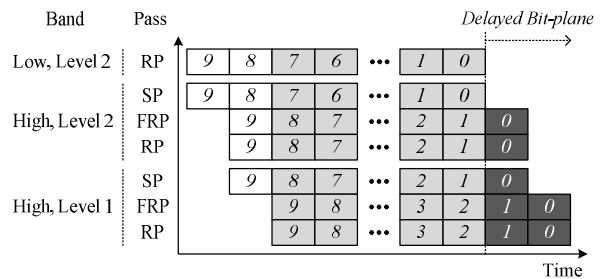


Fig. 2. Pipelining schedule of the high-throughput SPIHT encoder in [7] of which decomposition level is two.

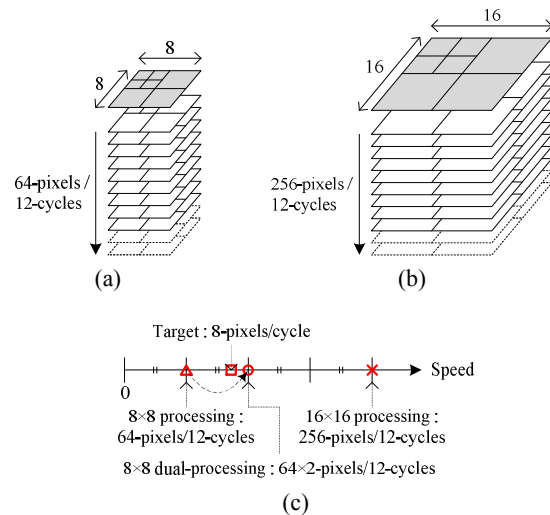


Fig. 3. Relationship between block size and throughput when the DWT decomposition level is two (a) processing an 8×8 block, (b) processing a 16×16 block, (c) target speed and processing speeds of (a) and (b).

presented in Fig. 2. In Fig. 3(a), input pixels with 8 bit-width are transformed into wavelet coefficients in the 2D DWT module and then transposed to 10 bit-planes in the transpose module. The transposed bit-planes are transferred to the 2D SPIHT modules. As shown in Fig. 2, the number of cycles to encode a single 8×8 block is 12 because delayed bit-planes, corresponded to the initial delayed two cycles, are added to avoid dependences. Therefore, the SPIHT hardware design shows throughput of 5.33 (=64/12) pixels/cycle when the block size used in FMC is 8×8, which is represented by a triangle mark in Fig. 3(c). In this case, the processing speed of SPIHT is slower than the target speed of 8 pixels/cycle which is represented by a square mark, and therefore the SPIHT design cannot work on the fly. Typically, the coding unit is a square-shaped block and its height and width values are determined in power of 2. When the height and width of an 8×8 block are doubled to increase the speed of

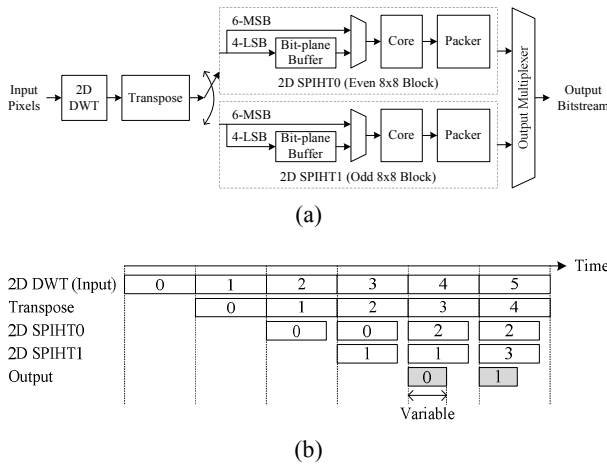


Fig. 4. The proposed encoder design with dual-core SPIHTs (a) block diagram, (b) timing diagram.

SPIHT, the block size becomes 16×16 . As shown in Fig. 3(b), the speed of SPIHT for a 16×16 block is 21.33 ($=256/12$) pixels/cycle, which is represented by a cross mark in Fig. 3(c). This speed is 2.67 times faster than the target speed. It leads to low hardware utilization because numerous hardware resources are exploited for the unnecessary speedup. If two 8×8 blocks are processed in parallel and pipelined manner, 128 ($=64 \times 2$) pixels are processed in 12 cycles. In this case, the SPIHT design has the throughput of 10.67 pixels/cycle, which is represented by a circle mark and becomes closer to the target speed of 8-pixels/cycle. As a result, this design is more efficient in hardware utilization, achieving the required processing speed.

Fig. 4(a) shows the block diagram of the proposed encoder design with two cores and the DWT decomposition level of 2. This design is composed of 2D DWT, transpose, 2D SPIHT0, 2D SPIHT1, and output multiplexer modules. The 2D SPIHT0 and 2D SPIHT1 modules process even and odd 8×8 blocks, respectively. Fig. 4(b) shows the timing diagram of this design. In this figure, the horizontal axis and the number in the rectangle represent execution time and the block number, respectively. Five stages shown in the vertical axis process coding blocks in pipelined manner. When input 8×8 block data come into the proposed encoder design, the 2D DWT module transforms the data into wavelet coefficients on the fly. The 2D DWT module transforms each coding block during one pipe time. The generated wavelet coefficients are then transposed to bit-planes in the transpose module. Note that the proposed design

utilizes single 2D DWT and transpose modules, although multi-core 2D SPIHT modules are used. Therefore, the overall hardware cost does not increase proportionally to the number of 2D SPIHT cores. When the pipe time for the transpose stage is finished, the 2D SPIHT module starts coding. The proposed SPIHT design is based on the parallel and pipelined method in [7]. As shown in Fig. 3, a single SPIHT module processes one bit-plane per cycle and shows the throughput of about 5.33 pixels/cycle for an 8×8 block, which is smaller than the target speed of 8 pixels/cycle. Therefore, bit-planes of input 8×8 block data are encoded during 2 pipe times. During the first pipe time, upper 6 bit-planes are encoded. On the other hand, lower 4 bit-planes and 2 delayed bit-planes are coded during the second pipe time. Therefore, the upper 6 bit-planes from the transpose module are directly transferred to the 2D SPIHT module, whereas the lower 4 bit-planes are stored in internal memory, denoted as a bit-plane buffer, and then transferred to the 2D SPIHT core at the next pipe time. In the proposed encoder design with dual-core SPIHT, two cores encode even and odd blocks in parallel. The bits generated in those cores are packed to the continuous bitstream in the packer module. As shown in Fig. 2, a single core is composed of many passes, and therefore the packer module includes logics to shift and merge the generated bits. Multi-cores simultaneously generate the packed bitstream. However, the bitstream of one core is only outputted through the output multiplexer because the completion times of the two cores are different. The generated bitstream is outputted after two pipe times in 2D SPIHT modules. The length of the outputted bitstream varies depending on the target CR.

Fig. 5 shows the processing orders of the proposed design using dual-cores and quad-cores with the block size of 8×8 and the DWT decomposition level of 2. The number in the rectangle represents the bit-plane number. Rectangles with numbers of -1 and -2 represent the delayed bit-planes, and 'B' represents a bubble cycle. Fig. 5(a) shows the processing order when dual-cores are used and eight pixels are inputted per cycle through a 64 bit-width bus. In this case, one pipe time is composed of eight cycles. As shown in this figure, during six cycles of the first pipe time, upper six bit-planes of the block0 are encoded in the core0. During six cycles of the next pipe time, lower four bit-planes and two delayed bit-planes of

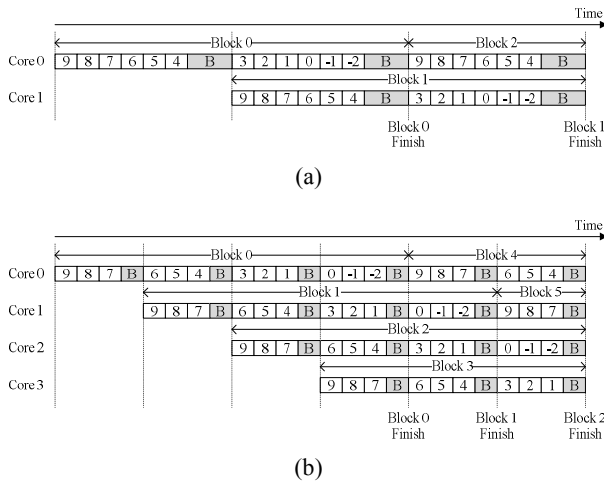


Fig. 5. Encoding process depending on the number of 2D SPIHT cores (a) dual-cores, (b) quad-cores.

the block0 are coded in the core0. At the same time, six bit-planes of the block1 are encoded in the core1. In other words, two blocks are simultaneously encoded since the first pipe time. The proposed scheme can be used for more cores. Fig. 5(b) shows the processing order when quad-cores are used. This situation occurs when 16 pixels come through a 128 bit-width bus per cycle and one pipe time for a single 8×8 block is composed of 4 cycles. During the first three pipe times, all cores do not work in parallel. However, quad-cores encode respective blocks in parallel after the first three pipe times as shown in Fig. 5(b).

Fig. 6(a) shows the block diagram of the proposed decoder design using the dual-core SPIHT. The decoder executes the opposite operation of the encoder. The design is composed of two inverse SPIHT modules, 2D iSPIHT0 and 2D iSPIHT1, the 2D inverse DWT module, and the output multiplexer module. The dual-core 2D iSPIHT modules, in common with the dual-core 2D SPIHT modules, work in parallel. The 2D iSPIHT module is composed of the pre-length calculation and initial address generator, which calculate bitstream addresses for each pass, the parser which uses those addresses and parses the bitstream, and the core which decodes the bitstream. In this module, about one bit-plane is processed per cycle. Reconstructed wavelet coefficients are stored in the coefficient buffer and then transferred to the 2D iDWT module. The 2D iDWT module executes the inverse DWT on the reconstructed wavelet coefficients and the final pixel data are

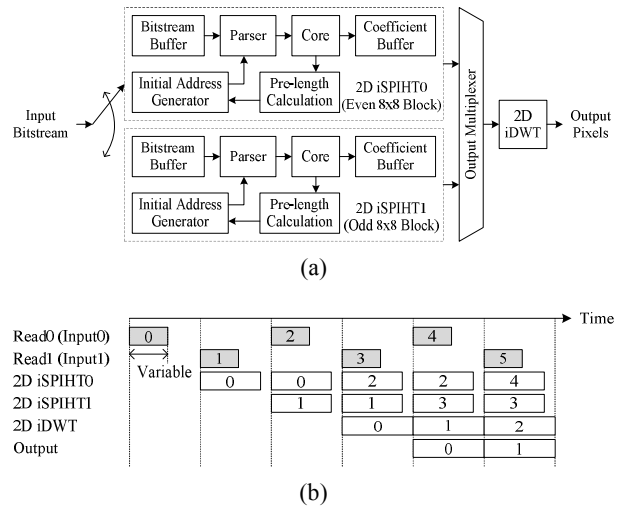


Fig. 6. The proposed decoder design with dual-core SPIHTs (a) block diagram, (b) timing diagram.

transferred to outside. Fig. 6(b) shows the timing schedule of the FMC decoder. In this figure, the horizontal axis and numbers in rectangles represent execution time and block numbers, respectively. One 2D iSPIHT module decodes a single 8×8 block during two pipe times and two 2D iSPIHT modules work in parallel. When the input bitstream for a single 8×8 block read is completed, the 2D iSPIHT module starts decoding at the next pipe time. The input bitstreams for even and odd blocks are alternately transferred to the 2D iSPIHT0 and 2D iSPIHT1 modules. After two pipe times, the wavelet coefficients are reconstructed. As completion times of two 2D iSPIHT modules are different, the reconstructed coefficients are, in turn, transferred to the 2D iDWT module through the output multiplexer. In the 2D iDWT module, each 8×8 block is processed during one pipe time. The final data from the 2D iDWT module are continuously outputted to outside.

2. Pass Reduction Scheme

The packer of encoder and the parser of decoder use shifters to pack and parse the bits, respectively. As the number of passes working in parallel increases, that of shifters used in packer and parser modules increases as well, which has a significant impact on the total hardware size. For example, a block with the decomposition level of 2 has seven bands, LL2, LH2, HL2, HH2, LH1, HL1, and HH1. The LL2 band has only

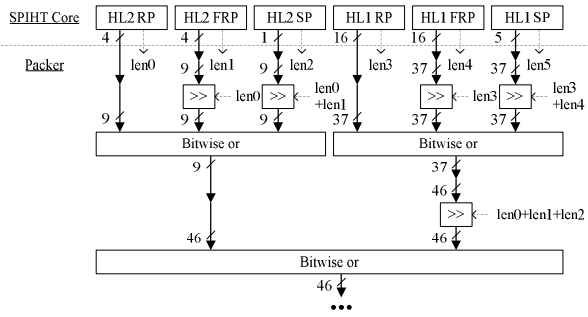


Fig. 7. Packing of HL coefficients in the previous method in which the size of block is 8×8 and the DWT decomposition level is two.

one pass, RP, while other six bands have three passes, SP, FRP, and RP. Therefore, nineteen passes operate in parallel in the FMC encoder. On the other hand, the FMC decoder has thirty-two passes which operate in parallel because the passes for sign bit decoding are added (RP of the LL2 band, and RP and FRP of other six-bands) [7]. Fig. 7 shows an example of the packing operation in the previous encoder design. Although this design simultaneously packs bits generated in nineteen passes, only six passes are shown in this example to reduce the space. When the block size is 8×8 , the maximum number of generated bits in HL2 RP, HL2 FRP, HL2 SP, HL1 RP, HL1 FRP, and HL1 SP are 4, 4, 1, 16, 16, and 5, respectively. Note that sign bits are not considered in this example. For packing bits in the HL1 band, bits in HL1 FRP are shifted by the number of bits in HL1 RP. Then, bits in HL1 SP are shifted by the total number of bits in HL1 RP and HL1 FRP. By using a bit-wise OR operator, all bits in the HL1 band are packed. Suppose that $\{1, 0, 1, 1\}$, $\{1, 1, 0, 1\}$, and $\{1, 0\}$ are generated in RP, FRP, and SP in the HL1 band, respectively. As the maximum number of bits in the HL1 band is 37 ($=16+16+5$), the bit-width of bits generated in respective passes is extended to 37. For example, bits in RP of the HL1 band become $\{1, 0, 1, 1, 0, \dots, 0\}$. As four bits are generated in HL1 RP, the bits in HL1 FRP are right-shifted by 4 and become $\{0, 0, 0, 0, 1, 1, 0, 1, 0, \dots, 0\}$. The bits in HL1 SP are right-shifted by 8 because total eight bits are generated in HL1 RP and HL1 FRP, and therefore the set $\{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, \dots, 0\}$ is generated. The 37 bit-sized three sets are merged by using an OR operator and the packed bitstream, $\{1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, \dots, 0\}$, is generated. By using the same manner, bits in the HL2 band are packed as well. When packing processes

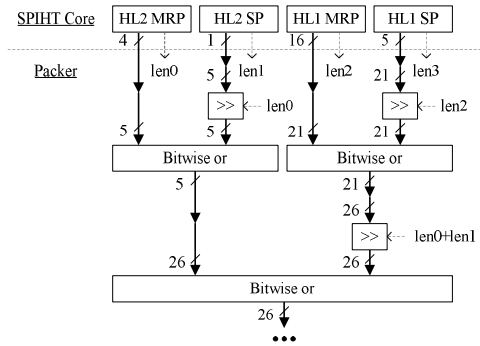


Fig. 8. Packing of HL coefficients in the proposed method.

for the HL1 and HL2 bands are finished, the bit-width of the packed bits is extended to 46. The generated bits in the HL1 band are right-shifted by the number of bits in the HL2 band, those two 46 bit-sized sets are then merged by using an OR operator. In the same manner, bits generated in all passes are packed. However, as the stage goes by, the bit size to merge increases. To this end, shifters with a large bit-width are required and therefore numerous hardware logics are exploited. Note that packer and parser modules of the SPIHT hardware design in [7] have 36.6% and 22.7% of encoder and decoder gate counts, respectively. Therefore, to reduce the total hardware cost, it is critical to reduce the numbers of passes and shifters.

This paper proposes a scheme to merge RP and FRP into a single pass called merged refinement pass (MRP) to reduce the number of passes. Previous RP and FRP execute respective functions on pixels with different present states. However, there is something that those two passes have in common. First, RP and FRP use the same threshold as shown in Fig. 2. Second, both of them apply the significance test on each pixel, not on a set. As one pixel is only tested in one pass depending on the present state, RP and FRP for the pixel do not work simultaneously. Therefore, those two passes are merged and the operation is determined depending on the present state. By using the proposed scheme, the number of encoder passes decreases from 19 to 13, and that of decoder passes decreases from 32 to 20.

Fig. 8 shows the modified logics when the proposed scheme is applied to the example in Fig. 7. As the number of passes decreases to two-thirds, that of shifters decreases as well. In addition, bit-widths of remained shifters are reduced. As a result, the size of SPIHT

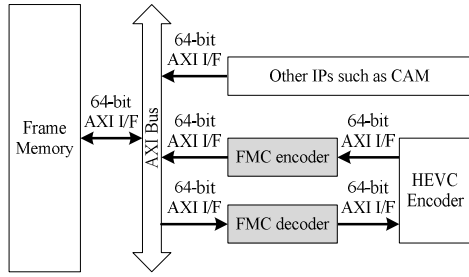


Fig. 9. Overall architecture.

hardware logics significantly decreases. Especially, the decrease of hardware logics is proportional to the number of multi-SPIHT cores. The proposed pass reduction scheme does not influence coding efficiency. However, the logic path in one pass becomes longer because the MRP logic is more complex than the RP or FRP logic. However, this path is not the critical path of the overall system, but rather the critical path, which is in the packer module, becomes shorter.

IV. INTEGRATION WITH HEVC ENCODER

This section presents the overall system when the proposed FMC hardware design is integrated with the AXI-based HEVC encoder. The proposed system compresses YUV 4:2:0 data with 8 bit-width. The bit-width of the AXI bus connected to the FMC encoder and decoder is 64 and the maximum burst length (BL) is 16. Fig. 9 shows the block diagram of the overall architecture. The FMC encoder and decoder are located between the HEVC encoder and the AXI bus, and communicate with the AXI bus using the same protocol which is previously utilized. The FMC encoder encodes data when the HEVC writes reconstructed frames to frame memory. On the other hand, the FMC decoder decodes the bitstream when the HEVC reads reference frames from the frame memory.

The FMC encoder receives 64 bits per cycle from the HEVC encoder. As the AXI bus uses a burst transferring protocol, 64 bits×BL-sized input data are continuous. The bitstream generated by the FMC encoder is transferred to frame memory using the BL determined depending on the target CR of the FMC. Fig. 10 shows examples of the changed BLs. In this example, the HEVC encoder sends 16 BL-sized data to the FMC encoder. As shown in Fig. 10(a), the 16 BL-sized data

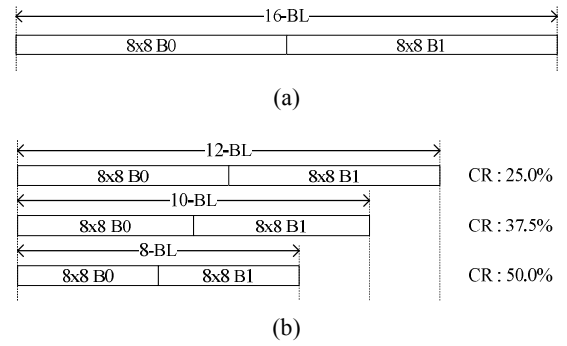


Fig. 10. Burst length of the single transfer (a) between FMC and HEVC, (b) between AXI bus and FMC.

correspond with two 8×8 block Y data. The CR is defined as

$$CR = (1 - \frac{\text{Total bits after compression}}{\text{Total bits before compression}}) \times 100. \quad (2)$$

Fig. 10(b) shows BLs when the encoded bitstream is transferred from the FMC encoder to frame memory through the AXI bus. BLs corresponding to target CRs of 25.0%, 37.5%, and 50.0% are 12, 10, and 8, respectively. It means that the bandwidth on the AXI bus, to which other IPs are also connected, is reduced. Note that compressed data may reduce the size of frame memory as well because the size of the encoded bitstream is fixed. When the FMC decoder receives read requests with the BL of 16 from the HEVC encoder, it transfers read requests with the changed BL depending on the target CR to frame memory. The bitstream from frame memory is decoded by the FMC decoder, and the decoded pixels are transferred to the HEVC encoder using the original BL, 16.

V. EXPERIMENTAL RESULTS

This section shows hardware implementation results of the proposed design. In addition, compression performance results are presented when the proposed design is integrated with HEVC.

1. Hardware Implementation

The proposed design codes an 8×8 block and the DWT decomposition level is set by 2. Input eight pixels come into the FMC encoder per cycle and the dual-cores

Table 1. Maximum throughput of the proposed SPIHT hardware design

Module		Maximum frequency (MHz)	Throughput (Mpixels/s)
Encoder	Total	556	4,448
	2D SPIHT Only	556	5,931
Decoder	Total	500	4,000
	2D iSPIHT Only	500	4,923

Table 2. Gate count and memory size of the proposed hardware design

Module		Gate count (Kgate)	Memory (bit)
Total		184.3	512
Encoder	Total	76.5	512
	2D DWT	12.5	-
	Transpose (Bit-plane Buffer)	13.8	(64×4)×2
	2D SPIHT	25.1	-
Decoder	Total	107.8	0
	2D iDWT	22.2	-
	2D iSPIHT	42.8	-

presented in Section III.1 are used. The proposed hardware design is targeted for ASIC implementation. To this end, the Verilog model is simulated and synthesized with the 65 nm TSMC technology library but without the place and route operations. Table 1 shows the maximum throughput results of the proposed hardware design. The first column represents respective FMC modules, and the second and third columns represent the maximum frequency and the maximum throughput, respectively. The maximum frequency of the FMC encoder is 556 MHz. As 64 pixels are processed in each pipe stage during 8 cycles, the maximum throughput is 4,448 Mpixels/s, which is shown in the second row of Table 1. However, when the 2D SPIHT hardware is only considered, 64 pixels are processed during 12 cycles and dual-cores are used. Therefore, the maximum throughput is 5,931 Mpixels/s as shown in the third row of Table 1. In the FMC decoder, the maximum frequency is 500 MHz, and therefore the maximum throughput is 4,000 Mpixels/s as shown in the fourth row. The processing cycle in the 2D iSPIHT is one cycle longer than that in the 2D SPIHT because the sign bit decoding is delayed [7]. Therefore, the maximum throughput of the dual-core 2D iSPIHT is 4,923 Mpixels/s, which is shown in the last row of Table 1.

Table 2 shows gate count and memory size results of the proposed hardware design. The gate count of the proposed encoder hardware is 76.5K, which accounts for

Table 3. Gate count change by the pass reduction scheme

Module		Gate count (Kgate)	
		w/o Pass reduction scheme [7]	w/ Pass reduction scheme
Single-core of 2D SPIHT	Total	29.7	25.1
	Packer	20.3	17.5
Single-core of 2D iSPIHT	Total	49.1	42.8
	Parser	14.2	11.2

41.51% of the total FMC gate count. The gate count of the single-core of 2D SPIHT is 25.1K and that of the dual-core hardware accounts for 65.62% of the overall encoder gate count. All internal memory in the encoder hardware is used for the bit-plane buffer. Along with the core, two same 256-bit memory elements are exploited for dual encoding. The gate count of the proposed decoder hardware is 107.8K, which accounts for 58.49% of the total FMC gate count. The reason why the gate count of the decoder is larger than that of the encoder is that the complexity of the 2D iDWT module is higher than that of the 2D DWT module and the number of passes in the 2D iSPIHT is larger than that in the 2D SPIHT. The proposed decoder uses dual-cores and the gate count of these modules accounts for 79.41% of that of the overall decoder hardware. As all buffers in the decoder hardware are implemented in registers, the total memory size of the decoder hardware is zero.

The gate count change results by the pass reduction scheme presented in Section III.2 is shown in Table 3. The second column shows gate count results when the proposed pass reduction scheme is not used (pass organization in [7] is used), while the third column shows gate count results when the proposed pass reduction scheme is used. In the single-core of 2D SPIHT, the proposed scheme reduces 4.6K gates which account for 15.49%. Especially, 2.8K gates are reduced in the packer module in which many shifters with a large bit-width are used. On the other hand, 6.3K gates which account for 12.83% are reduced in the single-core of 2D iSPIHT by the proposed scheme. In the parser module which is included in the single-core of 2D iSPIHT, 3.0K gates are reduced.

Table 4 shows hardware implementation results of the ‘16×8-single’ design, which is based on Kim *et al.* [7] and codes a 16×8 block, and the proposed design, which codes two 8×8 blocks using dual-cores. For fair comparison depending on block sizes, the proposed pass

Table 4. Comparison between single and dual-core SPIHT hardware designs

	16×8-single [7]		Proposed (8×8-dual)	
	enc.	dec.	enc.	dec.
Norm. throughput (pixel/cycle)	10.7	9.8	10.7	9.8
Gate counts (Kgate)	117.8	146.1	76.5	107.8
Gate counts/norm. throughput (Kgate×cycle/pixel)	11.0	14.9	7.1	11.0
Memory (bit)	512	0	512	0

reduction scheme presented in Section III.2 is applied to the 16×8-single design and the same synthesizing environment is used. The second row of Table 4 shows normalized throughput results of 2D SPIHT hardware designs. The 16×8-single design and the proposed design process the same number of pixels, 16×8 pixels and 8×8×2 pixels, and therefore throughputs of those designs are same. The third row shows gate count results. Gate counts of the 16×8-single encoder and decoder designs are 117.8K and 146.1K, respectively. On the other hand, gate counts of the proposed encoder and decoder are 76.5K and 107.8K, which are 41.3K and 38.3K smaller than those of the previous encoder and decoder designs, respectively. The fourth row shows results of which gate counts are divided by the normalized throughput of 2D SPIHT, which represents hardware area results normalized to the throughput. The results of the 16×8-single encoder and decoder are 11.0 and 14.9 Kgate×cycle/pixel, respectively, which are 1.55 times and 1.35 times larger than those of the proposed design. These results show that the proposed design is more efficient in hardware area than the previous design which simply increases the block size to improve throughput. Note that

the 2D DWT module in the 16×8-single design requires longer latency than the proposed design because the width of 16×8 block is larger than that of the proposed design. Moreover, the transpose module which transposes coefficients into bit-planes requires longer latency as well. The last row of Table 4 shows internal memory size results. The 16×8-single design and the proposed design use one 16×8×4-bit memory and two 8×8×4-bit memory, respectively, which means that the total internal memory sizes are same.

2. Compression Performance

This section presents compression performance results when the proposed method is applied to the HEVC. For the experiment, the HM13.0 reference software with the low delay P main configuration is used. As test sequences, two Class A (2560×1600), five Class B (1920×1080), and four Class E (1280×720) sequences with the 4:2:0 YUV format are used. In all test sequences, the number of frames is 30 and the first frame is I-frame while other frames are P-frames. Table 5 shows BDBR and BDPSNR results when MDA-SFL [6], 16×8-single SPIHT [7], and proposed algorithm are used as FMC. All algorithms use two target CRs, 25.0% and 50.0%. Results of the MDA-SFL in Table 5 are evaluated by using a software simulation. Originally, the MDA-SFL is a lossless coding algorithm, and therefore fixing the target CR is impossible. However, the iterative quantization method in [1] is applied to the MDA-SFL for comparison. In other words, when the bitstream length is larger than the TBL, a higher quantization level

Table 5. BDBR(%) and BDPSNR(dB) performance of the previous and proposed methods

Test sequence	BDBR (%) / BDPSNR (dB)					
	8×8 MDA-SFL [6]		16×8 single [7]		Proposed (8×8-dual)	
	CR = 25.0%	CR = 50.0%	CR = 25.0%	CR = 50.0%	CR = 25.0%	CR = 50.0%
Traffic	0.46 / -0.01	12.11 / -0.37	0.20 / -0.01	8.61 / -0.27	0.31 / -0.01	10.52 / -0.32
PeopleOnStreet	0.17 / -0.01	4.13 / -0.18	0.08 / 0.00	3.56 / -0.16	0.26 / -0.01	4.25 / -0.19
BQTerrace	2.11 / -0.05	23.56 / -0.47	1.37 / -0.03	13.02 / -0.25	1.83 / -0.04	15.39 / -0.30
BasketballDrive	-0.10 / 0.00	0.80 / -0.02	0.04 / 0.00	1.03 / -0.02	-0.09 / 0.00	1.73 / -0.04
Cactus	0.61 / -0.02	11.40 / -0.26	0.45 / -0.01	8.44 / -0.20	0.73 / -0.02	10.26 / -0.24
Kimono	-0.14 / 0.00	0.36 / -0.01	-0.42 / 0.01	0.54 / -0.02	-0.17 / 0.01	0.80 / -0.03
ParkScene	0.21 / -0.01	8.05 / -0.23	0.12 / 0.00	6.33 / -0.18	0.17 / -0.01	7.67 / -0.22
FourPeople	0.54 / -0.02	15.13 / -0.50	0.43 / -0.01	10.30 / -0.35	0.54 / -0.02	12.21 / -0.41
Johnny	0.15 / -0.01	12.10 / -0.28	-0.07 / 0.00	7.92 / -0.19	0.00 / 0.00	10.30 / -0.23
KristenAndSara	0.21 / -0.01	16.16 / -0.47	0.22 / -0.01	9.12 / -0.27	0.69 / -0.03	11.04 / -0.32
Vidyo1	0.23 / -0.01	12.42 / -0.37	0.23 / -0.01	8.22 / -0.25	0.53 / -0.02	10.58 / -0.32
Average	0.40 / -0.01	10.57 / -0.29	0.24 / -0.01	7.01 / -0.20	0.44 / -0.01	8.61 / -0.24

is applied to the input data. The number of iterations determined by this quantization method varies with block complexity, and thereby disturbing on-the-fly operation. For the on-the-fly operation, many cores with different quantization levels may be used in parallel but the hardware costs increase in proportion to the number of quantization levels. As shown in the second, fourth, and sixth columns of Table 5, the previous and proposed algorithms with the target CR of 25% show similar BDBR and BDPSNR results on average. However, when the target CR increases to 50%, the average BDBR of the proposed algorithm is 1.96% lower than that of the MDA-SFL as shown in the third and seventh columns. In addition, the average BDPSNR of the proposed algorithm is 0.05 dB higher than that of the MDA-SFL. These results represent that the proposed algorithm is more effective than the lossy MDA-SFL, especially, when the target CR is high. Compared to the 16×8 single SPIHT algorithm, the average BDBR of the proposed algorithm is 1.60% higher. It is because transform-based coding algorithms show higher coding efficiency as the size of coding block increases. However, the proposed design is more effective in hardware implementation given that the 16×8 single design requires larger hardware costs as shown in Table 4.

VI. CONCLUSIONS

This paper extends the high-throughput SPIHT hardware design in [7] to be fine-scalable and exploits it as an FMC integrated with the HEVC encoder. The proposed hardware design processes small-sized blocks in parallel and pipelined manner, resulting in similar coding efficiency and low hardware cost compared to the previous design with the same throughput. In addition, the proposed pass reduction scheme reduces hardware costs, which are particularly critical in packer and parser modules. The proposed design for a lossy FMC in video codecs shows higher coding efficiency than the previous FMC design, and it is more effective in video codec systems requiring limited frame memory size, burst transferring protocol, and data random access.

ACKNOWLEDGMENT

This research was supported by Basic Science

Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2015R1C1A1A0 2037625) and was supported by IDEC (EDA Tool, MPW).

REFERENCES

- [1] Y. Lee, C.-E. Rhee, and H.-J. Lee, "A New Frame Recompression Algorithm Integrated with H.264 Video Compression," *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2007.
- [2] S. Kim, D. Lee, H. Kim, N. X. Truong, and J.-S. Kim, "An Enhanced One-Dimensional SPIHT Algorithm and Its Implementation," *Displays Journal*, Vol. 40, pp. 68 - 77, Dec. 2015.
- [3] J. Kim and C.-M. Kyung, "A Lossless Embedded Compression Using Significant Bit Truncation for HD Video Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 20, No. 6, pp. 848-860, Jun. 2010.
- [4] T.-H. Tsai and Y.-H. Lee, "A 6.4 Gbit/s Embedded Compression Codec for Memory-Efficient Applications on Advanced-HD Specification," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 20, No.10, pp. 1277-1291, Oct. 2010.
- [5] Y. Jin and H.-J. Lee, "A Block-based Pass-parallel SPIHT Algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 22, No. 7, pp. 1064-1075, Jul. 2012.
- [6] L. Guo, D. Zhou, and S. Goto, "A New Reference Frame Recompression Algorithm and Its VLSI Architecture for UHD TV Video Codec," *IEEE Transactions on Multimedia*, Vol. 16, No. 8, pp. 2323-2332, Dec. 2014.
- [7] S. Kim, D. Lee, J.-S. Kim, and H.-J. Lee, "A High-Throughput Hardware Design of a One-Dimensional SPIHT Algorithm," *IEEE Transactions on Multimedia*, Vol. 18, No. 3, pp. 392-404, Mar. 2016.
- [8] S. Kim, D. Lee, J.-S. Kim, and H.-J. Lee, "A Block Truncation Coding Algorithm and Hardware Implementation Targeting 1/12 Compression for LCD Overdrive," *IEEE/OSA Journal of Display Technology*, Vol. 12, No. 4, pp. 376-389, Apr. 2016.
- [9] S. Kim, M. Kim, J.-S. Kim, and H.-J. Lee, "Fixed-

Ratio Compression of an RGBW Image and Its Hardware Implementation,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 6, No. 4, pp. 484-496, Dec. 2016.

- [10] S. Kim and H.-J. Lee, “RGBW Image Compression by Low-Complexity Adaptive Multi-Level Block Truncation Coding,” *IEEE Transactions on Consumer Electronics*, Vol. 62, No. 4, pp. 412-419, Nov. 2016.
- [11] A. Said and W. 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, Jun. 1996.
- [12] F. W. Wheeler and W. A. Pearlman, “SPIHT Image Compression Without Lists,” *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 6, pp. 2047-2050, Jun. 2000.
- [13] T. W. Fry and S. A. Hauck, “SPIHT Image Compression on FPGAs,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 9, pp. 1138-1147, Sep. 2005.
- [14] P. Corsonello, S. Perri, G. Staino, M. Lanuzza, and G. Cocorullo, “Low Bit Rate Image Compression Core for Onboard Space Applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 1, pp. 114-128, Jan. 2006.
- [15] C.-C. Cheng, P.-C. Tseng, C.-T. Huang, and L.-G. Chen, “Multi-mode Embedded Compression Codec Engine for Power-Aware Video Coding System,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 19, No. 2, pp. 141-150, Feb. 2009.



Sunwoong Kim received B.S., M.S., and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 2010, 2012, and 2016, respectively. He is currently working as a postdoctoral researcher

at Seoul National University. His research interests are hardware design for multimedia systems, computer architecture, and memory systems.



Ji Hun Jang received B.S. degree in Information and Communication Engineering from Inha University, Incheon, Korea, in 2015. He is currently a master student in Department of Information and Communication Engineering at Inha University, Incheon, Korea. His research interests are VLSI design of video coding and virtual reality.



Hyuk-Jae Lee received B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Korea, in 1987 and 1989, respectively, and obtained a Ph.D. degree in Electrical and Computer Engineering from Purdue University at West Lafayette, Indiana, in 1996. From 1998 to 2001, he worked at the Server and Workstation Chipset Division of Intel Corporation in Hillsboro, Oregon as a senior component design engineer. From 1996 to 1998, he was on the faculty of the Department of Computer Science of Louisiana Tech University at Ruston, Louisiana. In 2001, he joined the School of Electrical Engineering and Computer Science at Seoul National University where he is currently working as a Professor. His research interests are in the areas of computer architecture and SoC design for multimedia applications.



Chae Eun Rhee received the B.S., M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 2000, 2002 and 2011, respectively. From 2002 to 2005, she was with the Digital TV Development Group, Samsung Electronics Company Ltd., Suwon City, Korea, as an Engineer, where she was involved in bus architecture and MPEG decoder development. In 2013, she joined the Department of Information and Communication Engineering at Inha University, Korea, where she is currently working as an associate professor. Her research interests include algorithm and architecture design of video coding for HEVC and H.264/AVC, configurable video coding for real time systems and the next generation virtual reality systems.