Fine Granular Scalable Video Coding Using Context-Based Binary Arithmetic Coding for Bit-Plane Coding

Seung-Hwan Kim, *Student Member, IEEE*, and Yo-Sung Ho, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a new efficient entropy coding scheme, namely context-based binary arithmetic coding for bit-plane coding (CBACBP), for the enhancement layer of fine granular scalable (FGS) video coding. The basic structure of proposed CBACBP is based on the traditional bit-plane coding of MPEG-4 FGS. However, in order to enhance the coding efficiency of bit-plane coding, a newly designed context-based binary arithmetic coding scheme is used. In CBACBP, we apply three types of probability estimation techniques. The first type relies on local information of a given symbol to code, such as bit-plane level, data level and frequency component. In the second type, the preceding symbols in the same and/or higher bit-planes are used. In the third type, we consider complexity of a block including the symbol based on the number of coded nonzero coefficients in the block. Experimental results show that the proposed CBACBP improves the PSNR up to 1.2 and 0.5 dB compared with MPEG-4 FGS and JSVM, respectively.**

***Index Terms*—Bit-plane coding, fine granular scalable (FGS), scalable video coding, signal-to-noise ratio (SNR) scalability.**

I. Introduction

WITH the growth of the transmission of multimedia content on the Internet, scalable video coding (SVC) has actively been researched for the network video application, such as Internet streaming video [1]. Considering the requirements for the Internet streaming video application, fine-granular-scalability (FGS) video-coding scheme is introduced originally in [2]. In the initial stage of research on FGS in MPEG-4, three outstanding approaches were proposed for coding the FGS enhancement layer: wavelet, DCT, and matching-pursuit based methods [3]. In particular, the performance of different variations of bit-plane DCT-based coding [4], [5] and wavelet compression methods have been studied, compared, and presented in [6]. Based on a thorough analysis of the FGS enhancement-layer (signal-to-noise ratio (SNR)) signal, the study in [6] concluded that both bit-plane DCT coding and embedded zero-tree wavelet (EZW)-based compression provide very similar results. The same conclusion was reached by the MPEG-4 FGS effort. Consequently, due to the fact that the FGS base-layer is coded using MPEG-4 compliant DCT coding, employing embedded DCT method for compressing the enhancement layer is a sensible option [6].

Therefore, the basic MPEG-4 FGS coding scheme is built upon the original FGS scalability structure proposed in [2], and embedded DCT coding of the enhancement layer as proposed in [4] and [5]. An overview on the FGS video coding technique defined in MPEG-4 is provided in detail in [7].

With the introduction of H.264/AVC [8] the most powerful and state-of-the-art standard, a new FGS video coding scheme called AVC-FGS was proposed in [9]. AVC-FGS uses a H.264/AVC (JM 5.0) for base layer and UVLC for the FGS enhancement layer. The study showed that a coding efficiency of UVLC was slightly inferior to MPEG-4 FGS. In the study in [10], the variable block-size transform and context-based entropy coding techniques were designed for the enhancement layer of FGS video coding. In particular, in order to take the advantage of the characteristics and correlations of symbols coded in the FGS enhancement layer, simple context models are designed for the arithmetic coding according to symbol type and transform size. In the study in [11], a context adaptive bit-plane coding (CABIC) with a stochastic bit reshuffling (SBR) scheme was proposed. In order to improve coding efficiency, CABIC constructs context models based on both the energy distribution in a block and the spatial correlations in the adjacent blocks. Moreover, it exploits the context across bit-planes to save side information.

Recently, the scalable extension of H.264/AVC was selected as the first Working Draft [12] and a reference encoder was described in the joint scalable video model (JSVM 0) [13]. In order to support fine granular SNR scalability, progressive refinement (PR) slice [14]–[16] is introduced to JSVM. A refinement signal that corresponds to a bisection of the quantization step size is represented in PR slice. In addition, with the exception of the modified coding order, the CABAC [17] entropy coding specified in H.264/AVC is reused in PR slice. According to SVC working draft [18] and JSVM annex S [19], key picture uses the conventional closed-loop motion compensation and the reference pictures for the key picture shall be base representation without FGS enhancement. On the contrary, the reference pictures of nonkey picture shall be base representation + FGS enhancement. Hence, if FGS layers are truncated drift error is inevitable for the nonkey pictures. In order to limit the corresponding drift, temporal prediction in a key picture is limited to the base layer. Thus, the key picture is used as resynchronization point between encoder and decoder. In the meantime, [27]

Manuscript received June 20, 2006; revised January 1, 2007. This work was supported in part by MIC through RBRC at Gwangju Institute of Science and Technology (GIST) and by MOE through the BK21 project. This paper was recommended by Associate Editor M. van der Schaar.

The authors are with the Department of Information and Communications, Gwangju Institute of Science and Technology (GIST), Gwangju 500-712, Korea (e-mail: kshkim@gist.ac.kr; hoyo@gist.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2007.903778

TABLE I
BASIC CODING STRUCTURES FOR IMPLENENTED FGS CODECS

Codec	Base	Enhancement
H.264-FGS-VLC	H.264/AVC	MPEG-4FGS
H.264-FGS-CBACBP	H.264/AVC	CBACBP

proposed a leaky prediction based solution to improve the FGS coding of the close-loop P frames by using temporal prediction signal which is adaptively formed from both the enhancement layer reference frame and base layer reference frame based on the information coded in the base layer. This solution is referred to as AR-FGS (FGS coding with adaptive reference). AR-FGS significantly improves the FGS layer coding efficiency with effective control on the drift.

In this paper, we focus on the design of a new efficient FGS coding scheme and propose context-based binary arithmetic coding for bit-plane coding (CBACBP). The proposed CBACBP seems to be similar to [10] and [11] in the aspect of using context-based arithmetic coding method for FGS coding. However, one of the main differences is what context model to use and how to use it. In CBACBP, we have adopted three types of probability estimation techniques, such as local information, the preceding symbols in the same and/or higher bit-planes and the statistical distribution of coded symbol according to block. In Section III, we will explain these in detail.

The coding structure of CBACBP is also quite different from that of JSVM FGS. CBACBP adopt the structure of traditional bit-plane coding in [7] but JSVM FGS combine two structures: the general layered concept of requantization and progressive refinement coding for a given requantized layer. Therefore, CBACBP and JSVM FGS use different techniques of context modeling because they adopt different coding structures.

In order to verify the coding efficiency of the proposed method, we have implemented two kinds of FGS codecs: H.264-FGS-VLC and H.264-FGS-CBACBP. Table I shows the basic coding structures for those codecs. In H.264-FGS-VLC, H.264/AVC (JM 9.5) [20] is used for the base layer coding and the traditional bit-plane coding in MPEG-4 FGS is used for the enhancement layer coding. In H.264-FGS-CBACBP, we also use H.264 (JM9.5) for the base layer coding and the proposed CBACBP scheme is used for the enhancement layer coding.

The rest of this paper is organized as follows. Section II briefly reviews the two FGS coding schemes in MPEG-4 FGS and JSVM, respectively. In Section III, we propose an efficient FGS coding scheme named CBACBP. In Section IV, the performance of the proposed coding technique is compared with JSVM and MPEG-4 FGS. Finally, the paper is completed with some conclusions in Section V.

II. Overview of Previous FGS Coding

In this section, we will discuss two kinds of well-known FGS coding techniques used in MPEG-4 FGS and JSVM, respectively.

A. *H.264 Based MPEG-4 FGS*

In MPEG-4 FGS, MPEG-4 advanced simple profile (ASP) is used for the base layer coding. So, the replacement of MPEG-4

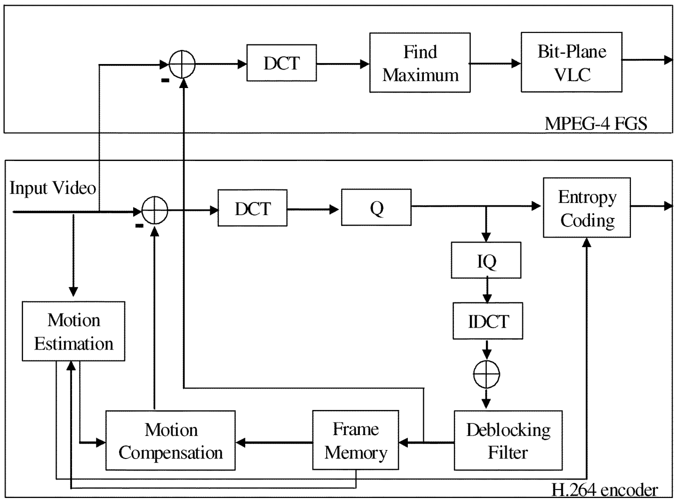

Fig. 1. Structure of H.264-FGS-VLC.

ASP with H.264/AVC for the base layer coding is expected to bring improvements in coding efficiency. There has been some works reported regarding the combination of the MPEG-4 FGS and the H.264/AVC [9], [21]. The approach taken is a direct implementation of the MPEG-4 FGS enhancement layer, without any modification, on the top of the H.264/AVC encoder. The base layer is implemented by using H.264/AVC reference software version JM 9.5. In the enhancement layer, the bit-plane coding method in MPEG-4 FGS is directly implemented. Fig. 1 shows our designed H.264-FGS-VLC encoder.

The bit-plane coding in MPEG-4 FGS (or H.264-FGS-VLC) considers each quantized DCT coefficient as a binary number of several bits instead of a decimal integer of a certain value [7]. For each DCT block, the 64 absolute values are zigzag ordered into an array. A bit-plane of the block is defined as an array of 64 bits, taken one from each absolute value of the DCT coefficients at the same significant position. For each bit-plane of each block, (RUN, EOP) symbols are formed and variable-length coded ("Bit-plane VLC") to produce the output bitstream.

Bit-plane VLC is based on Huffman coding. Since the statistics of the first three bit planes (MSB, MSB-1, and MSB-2) are very different from each other and from the lower bit-planes, corresponding four VLC tables have been designed for MSB plane, MSB-1 plane, MSB-2 plane, and the other bit planes.

The designed bit-plane VLC tables reflect the statistical distribution of each bit plane. But still there is plenty of room for improvement. It still shares the fundamental disadvantage of Huffman coding that assigns a codeword containing an integral number of bits to each symbol. The usage of only fixed VLC tables does not allow an adaptation to the actual symbol statistics, which may vary over spatial and temporal as well as for different source material and coding conditions. In MPEG-4 FGS [7], no temporal prediction at all is performed for the enhancement layers because temporal prediction in the enhancement layer causes drift problem.

B. *JSVM*

SVC is an extension of the newly adopted H.264/AVC video standard. The basic design of SVC can be classified as lay-

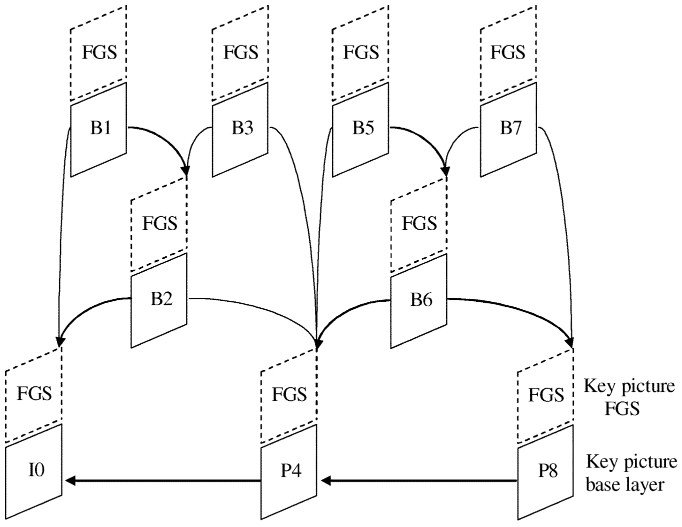

Fig. 2. Structure of hierarchical B-pictures using FGS.

ered video codec. In order to support fine granular SNR scalability, JSVM adopted so-called progressive refinement (PR) slices [14]–[16]. Each PR slice is regarded as a FGS layer which represents a refinement signal that corresponds to a bisection of the quantization step size (QP increase of 6). Each FGS layer is represented as a group of multiple bit-planes. However, these bit-planes are coded by a cyclical block coding [22], [23] instead of traditional bit-plane coding used in MPEG-4 FGS. The coding order of transform coefficient levels has been modified. Instead of scanning the transform coefficients macroblock by macroblock as it is done in "normal" slices, the transform coefficient blocks are scanned in several paths, and in each path only a few coding symbols for a transform coefficient block are coded. So, the quality of the SNR base layer can be improved in a fine granular way. Therefore, with the exception of the modified coding order, the CABAC entropy coding as specified in H.264/MPEG4-AVC [17] is reused.

As we mentioned in the Section I, the reference pictures of nonkey picture shall be base representation + FGS enhancement. Fig. 2 shows the typical example of the reference pictures for the key picture and nonkey picture. As shown in Fig. 2, the reference pictures for key pictures are the base representations, which mean the decoded picture without using FGS enhancement, and those of nonkey pictures are base quality + FGS enhancement [24]–[26].

When FGS layers are truncated the reference pictures that are used in encoder and decoder are different. In order to limit the corresponding drift, the key pictures are used as resynchronization point between encoder and decoder as illustrated in Fig. 2.

III. Proposed FGS Coding Algorithm

In this section, we propose a new efficient FGS coding scheme called context-based binary arithmetic coding for bit-plane coding (CBACBP). In Section III-A, we introduce the basic coding structure of CBACBP and CBP (coded block pattern) coding. The procedure of CBP coding is depicted in Fig. 3. In Section III-B, probability estimation technique in CBACBP is discussed.

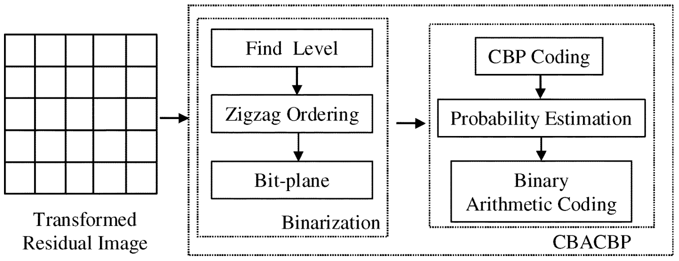

Fig. 3. Basic coding structure of CBACBP.

A. Overall Coding Structure of CBACBP

Variable length coding (VLC) assigns a variable length code word to each codeword based on the probability of its occurrence. In the MPEG-4 FGS, Huffman coding is used for the enhancement layer coding. Huffman coding can be optimum if the symbol probability is an integer power of 1/2. However, arithmetic coding is a data compression technique that encodes data by creating a code string, which represents a fractional value on the number line between 0 and 1. Therefore, it actually achieves the theoretical entropy bound of compression efficiency for any source.

Therefore, in order to design an efficient FGS coding scheme, we propose context based binary arithmetic coding for bit plane coding. The proposed CBACBP adopt arithmetic coding and utilizes many useful contexts to get a skewed probability distribution, which hopefully gives high probability to the symbols that actually occur. We can thus enhance the coding efficiency of binary arithmetic coding. We will discuss more details in Section III-B.

Fig. 3 shows the basic coding structure of CBACBP. The basic structure of the proposed CBACBP is based on the traditional bit-plane coding (bit-plane VLC) in MPEG-4 FGS. However, the bit-plane VLC is replaced with CBACBP to enhance the coding efficiency of the bit-plane coding. In Fig. 3, the residual signal obtained from subtracting reconstructed-based image from original image is divided into 4×4 blocks and transformed. After that, those DCT coefficients are zigzag ordered and formed into several bit-planes according to their magnitude levels.

In the CBP coding, we encode whether the MSB plane of each block is reached or not. In the binarization process depicted in Fig. 3, the MSB plane of each 4×4 block can vary from one block to another. Hence, before the MSB plane in each block is reached, we need to send (code) a signal, msb_not_ reached, which indicates whether MSB plane is reached or not. In order to signal msb_not_ reached, we utilize a similar coding scheme used in MPEG-4 FGS [7]. From extensive experiments, we have confirmed that the CBP coding in MPEG-4 FGS provided very good coding efficiency. Since many 4×4 blocks have fewer bit planes than the maximum number of bit planes in a frame, there are many cases that MSB is not reached. Therefore, to signal msb_not_reached for every block is not efficient. Hence, we classify each macroblock into three types of blocks, such as 4×4, 8×8, and 16×16 blocks according to size. With the same methodology used in MPEG-4 FGS, "MB_msb_not_reached" for each block is hierarchically coded by the unit of MB. By combining FLC and VLC in MPEG-4 FGS, we encode MB_msb_not_reached and msb_not_reached. More details on CBP coding and binarization processes are explained in [7].

In the probability estimation step, we estimate the probability of a given symbol to code. Based on the context models, a probability model is selected such that the corresponding choice may depend on previously encoded binary symbols. In the binary arithmetic coding step, the binary arithmetic coding engine makes a bitstream from the probability estimation of a given symbol.

B. Probability Estimation

Binary arithmetic coding is based on the principle of recursive interval subdivision and an estimated probability of the given symbol is represented by its range. The given interval is subdivided into two subintervals, LPS and MPS. Depending on the observed binary decision, either identified as the LPS or the MPS, the corresponding subinterval is then chosen as the new current interval. Thus, if we know the probability of a given symbol to be zero, $P(0)$, the probability of the symbol to be one is automatically determined by $1-P(0)$. Therefore, in the probability estimation step, only the estimation of $P(0)$ is necessary.

In order to efficiently estimate the probability of a given symbol to code, we apply three types of probability estimation techniques. The first type relies on local information, such as bit-plane level and frequency component, of the symbol. The second type mainly depends on the preceding symbols in the same bit-plane layer as well as in the different bit-plane layers. In the third type, we use complexity of a block where a current symbol is included. The complexity of a block is measured by the number of nonzero coefficients. In the following paragraphs, we discuss these three types of probability estimation techniques in detail.

In the bit-plane coding, we generally have different statistical distribution of binary symbols for each bit-plane level. In order to more accurately estimate $P(0)$ for a symbol, we basically use the local information. Local information includes the bit-plane level, the data level and the scanning position. Each of them is defined as follows.

BPL (bit-plane level)	The index of the bit-plane layer of a given symbol in terms of the most significant bit-plane level in each frame.
DL (data level)	The index of the bit-plane layer of a given symbol in terms of the most significant bit-plane level in each 4 × 4 block.
SP (scanning position)	The order of the zigzag scanning of a given symbol in each 4 × 4 block (0 ∼ 15).

Fig. 4 represents the local information. In Fig. 4, the frame is represented by 6 bit-plane levels from 0 to 5 and the block has 5 bit-plane levels. Hence, in the block, the value of *BPL* is always larger than that of *DL* by one. The binary symbol "A" in Fig. 4 has local information of $BPL = 3$, $DL = 2$, and $SP = 3$.

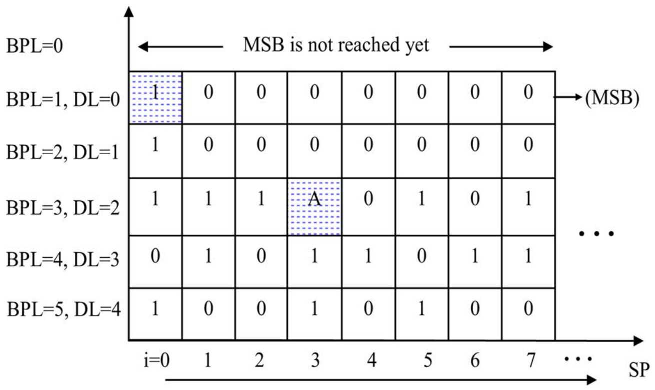

Fig. 4. Local information of a given symbol in a block.

TABLE II
PROBABILITY DISTRIBUTION OF SYMBOL ZERO ACCORDING TO THE "*BPL*" AND "*DL*"

DL \ *BPL*	0	1	2	3
0(MSB)	0.983	0.973	0.964	0.959
1(MSB-1)	-	0.964	0.914	0.906
2(MSB-2)	-	-	0.913	0.837
3(Others)	-	-	-	0.838

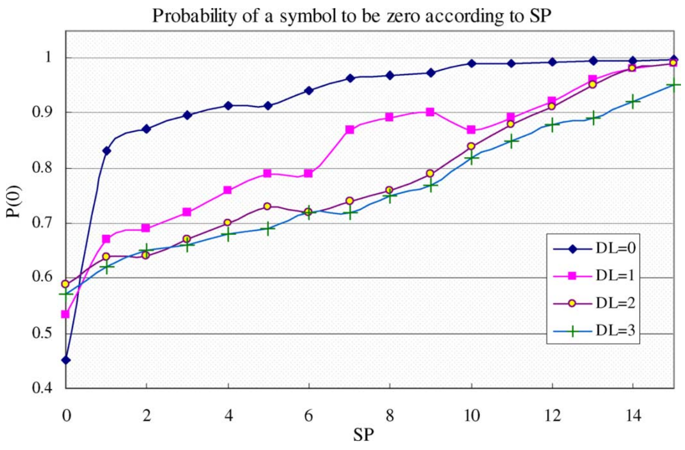

Fig. 5. Probability of a symbol to be zero according to *SP*.

Not every 4 × 4 DCT block has the same number of bit-planes. In other words, the MSB plane of each individual block can vary from block to block in terms of *BPL*. Hence, for a given data level, the statistical distribution of binary data needs to be determined by considering corresponding *BPL*. Based on the extensive experiments of many video sequences, we have observed the probability distribution of binary data according to *BPL* and *DL*. In Table II, the probability distribution of zeros is represented. We can see that the probability of zero is changing according to not only *DL* but also *BPL*. Hence, if we use *DL* and *BPL* then, we can get more skewed probability distribution than only use *DL*. Fig. 5 represents the probability distribution of the symbol zero according to *SP*. In Fig. 5, we observed that probability distribution of the binary data is variable according to *SP*.

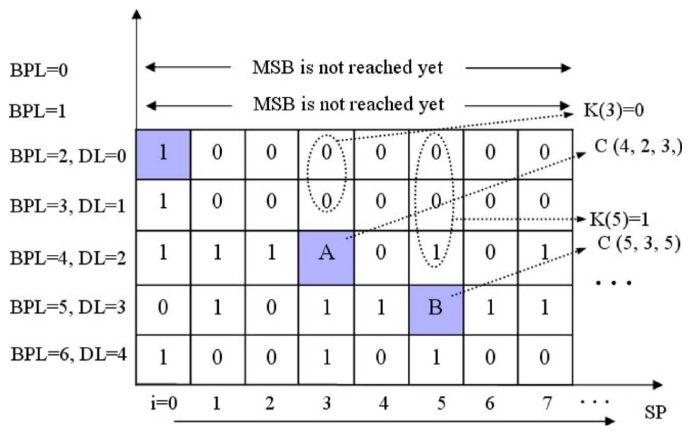

Fig. 6. Contexts of a given symbol in a block.

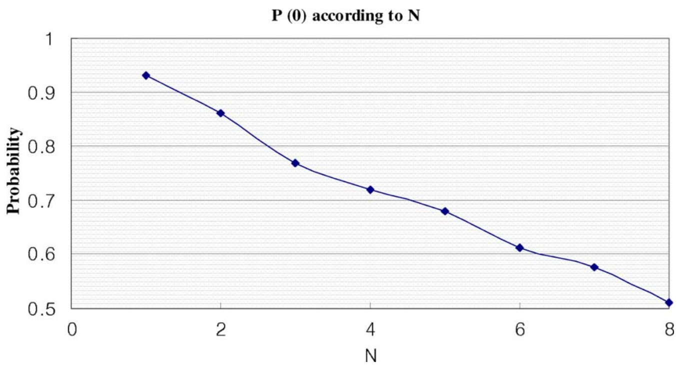

Fig. 7. Variation of $P(0)$ according to N.

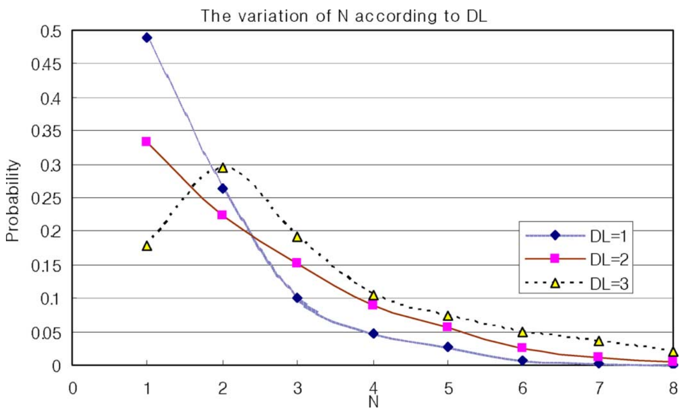

Fig. 8. The variation of N according to *DL*.

Context modeling is based on the fact that a symbol which has appeared recently will appear in a near future, so the more times that it appears the higher probability that it appears again. We use context because the probability of a binary symbol appearing in a given position heavily depends on the symbols that have appeared before. When using the information of context we also have to take care of what is the context of the current symbol, and thus we take only the probability of the symbols which have appeared under the given context. In bit-plane coding, binary data are context sensible, that is, under a given context the same symbols tend to appear, so we can get a more reliable probability. The goal of a context modeling is to get a skewed probability distribution, which hopefully gives high probability to the symbols which actually occur, and thus we can code them with fewer bits. As shown in Fig. 6, we define several terminologies as follows.

$C(BPL, DL, SP)$	Represents a given binary symbol in a 4×4 block.
$K(SP)$	The index that represents whether nonzero symbol ("1") has been already reached or not at a given *SP* in a block. It is defined as

$$K(SP) = \begin{cases} 0, & \text{for } \sum_{DL=0}^{\mathrm{L}-1} C(BPL, DL, SP) = 0 \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

We once define a block as a complex block, which includes many nonzero coefficients. In the complex block, we have more nonzero symbols compared to a normal block. Hence, for a complex block, we need to use a different probability estimation model. In order to efficiently apply the concept to the bit-plane coding, we define "complex-bit-plane" which has more nonzero symbols than in the normal bit-plane level. Therefore, the index N is used to measure complexity of the corresponding bit-plane and obtained by using $K(SP)$ as follows:

$$N = \sum_{SP=0}^{63} K(SP). \quad (2)$$

Then, in a given bit-plane, we determine whether the corresponding bit-plane is "complex-bit-plane" or not. For complexity-bit-plane, we consider four cases according to the value of N as

$$\begin{aligned} A \le N &< A + \alpha \\ A + \alpha \le N &< A + 2\alpha \\ A + 2\alpha \le N &< A + 3\alpha \\ A + 3\alpha \le N & \end{aligned} \quad (3)$$

where A represents the threshold value to determine whether the corresponding bit-plane is "complex-bit-plane" or not and α is a value used to classify the "complex-bit-plane" into four cases according to the range of N. For simplicity, in our experimental results, we fixed A into 4, 5, and 6 according to $DL = 1$, $DL = 2$, and $DL = 3$, respectively. The value of α is set to one.

The variation of $P(0)$ according to N and the variation of N according to *DL* are depicted in Figs. 7 and 8, respectively.

C. Context Index Modeling

In this subsection, we give a shape to a probability estimation using the context modeling mentioned in Section III-B. The entity of probability models used in CBACBP can be represented by such four context arrangement functions. Hence, each probability model can be identified by a unique so-called context index C as

$$C = F_{\Delta}(DL, BPL, N) + F_{\theta}(N) + F_{\delta}(DL, N, SP) + F_{\chi}(DL, N, K(SP)). \quad (4)$$

TABLE III
CONTEXT DATA AND ASSOCIATED CONTEXT CATEGORY

BPL	*DL*	$F_\Delta(DL, BPL, N)$ (*context category*)
0	0	0
1	0	1
	1	2
2	0	2
	1	3
	2	3
3	0	2
	1	3
	2	4
	3	4
'*Complex-bit-plane*'		5

The first context arrangement function $F_\Delta(DL, BPL, N)$ indicates a specified context category of the corresponding context index. We have classified the context model into six categories. The first five categories are differentiated based on *BPL* and *DL*, and the sixth category is assigned to the case of a "complex-bit-plane." Table III represents the context categories according to context models.

The second context arrangement function $F_\theta(N)$ indicates the context offset for the last context category ("complex-bit-plane"). In our CBACBP, $F_\theta(N)$ can be represented by one of four values (0, 1, 2, and 3) according to the range of N defined in (3). The third context arrangement function, $F_\delta(DL, N, SP)$, represents the context offset according to *SP*. $F_\delta(DL, N, SP)$ is classified into three types of models according to *DL* and *context category*. Specifications for the third context offset according to *SP* are explained as follows.

```
If ( $F_\Delta(DL, BPL, N) < 5$)
{
  If ($DL == 0$)
    $F_\delta(DL,N,SP) = \begin{cases} SP, & \text{if } 2 > SP \\ 2 + 4\cdot(SP-2), & \text{if } 2 \le SP \le 11 \\ 38 + (SP-11), & \text{if } 11 < SP \end{cases}$
  else
    $F_\delta(DL,N,SP) = \begin{cases} 2\cdot SP, & \text{if } 2 > SP \\ 4 + 5\cdot(SP-2), & \text{if } 2 \le SP \le 11 \\ 49 + 2\cdot(SP-11), & \text{if } 11 < SP \end{cases}$
}
else $F_\delta(DL,N,SP) = 2\cdot SP$.
```

In Table IV, we represent the second and third context offsets together by considering *DL* and context category in terms of a given *SP*.

Finally, the fourth context arrangement function, $F_\chi(DL, N, K(SP))$, represents context increment to finally obtain the corresponding context index C based on the previously obtained context category and context offsets. In particular, in order to take the advantage of the characteristics and correlations of symbols coded in the neighboring *SP*, simple context models are designed as

$$M(SP) = 2\cdot C(BPL, DL, SP-1) + C(BPL, DL, SP-2). \tag{5}$$

The context increment index is specified as follows.

```
If ($F_\Delta(DL, BPL, N) < 5$)
{
  If ( $2 \le SP < 11$)
  {
    $F_\chi(DL, N, K(SP)) = \begin{cases} M(SP), & \text{if } DL=0 \\ M(SP)+4\cdot K(SP), & \text{if } DL>0 \end{cases}$
  }
  else
    $F_\chi(DL, N, K(SP)) = \begin{cases} 0, & \text{if } DL = 0 \\ K(SP), & \text{if } DL > 0 \end{cases}$
}
else $F_\chi(DL, N, K(SP), SP) = K(SP)$.
```

TABLE IV
BASIC CODING STRUCTURES FOR IMPLENENTED

SP	*DL*= =0	*DL* > 0	$F_\Delta(DL, BPL, N) = 5$
0	0	0	0
1	1	2	2
2	2	4	4
3	6	9	6
4	10	14	8
5	14	19	10
6	18	24	12
7	22	29	14
8	26	34	16
9	30	39	18
10	34	44	20
11	38	49	22
12	39	51	24
13	40	53	26
14	41	55	28
15	42	57	30

Therefore, we represent the relation between context index and context argument functions in Table V. Basic procedure of the probability estimation technique in CBACBP is represented in Fig. 9. In Fig. 9, "BAC" represents the binary arithmetic coding engine that generates a bitstream from $P(0)$ of the corresponding symbol. In order to implement binary arithmetic coding, we followed the general rule for the arithmetic coding. At the start, the coding interval is initialized to the whole scale $[0, T)$. We then project the estimated probability model of a new symbol to the initialized interval and obtain a new interval. The

TABLE V
CONTEXT INDEX AND CONTEXT ARGUMENT FUNCTIONS

Context category: $F_{\Delta}(DL, BPL, N)$	0	1	2	3	4	5			
Context offset: $F_{\theta}(N)$	0					$F_{\theta}(N)=1$	$F_{\theta}(N)=2$	$F_{\theta}(N)=3$	$F_{\theta}(N)=4$
						0	32	64	96
Context offset: $F_{\delta}(DL, N, SP)$	$F_{\delta}(DL,\ N, SP)$					$2 \cdot SP$			
Context increment: $F_{\chi}(DL, N, K(SP))$	$F_{\chi}(DL, N, K(SP))$					$K(SP)$			

TABLE VI
COMPARISON OF CODING BIT RATE BETWEEN CBACBP AND H.264-FGS-VLC

Sequences	Coded Bit-planes	H.264-FGS-VLC (kbps)	CBACBP (kbps)	Bit-Savings (%)
FOREMAN	Base+2bit-planes	317	282	11.04
	Base+4bit-planes	3033	2643	12.85
BUS	Base+2bit-planes	652	593	9.04
	Base+4bit-planes	4809	4314	10.29
MOBILE	Base+2bit-planes	845	740	12.42
	Base+4bit-planes	6326	5487	15.33
FOOTBALL	Base+2bit-planes	643	541	15.86
	Base+4bit-planes	5107	4350	17.15

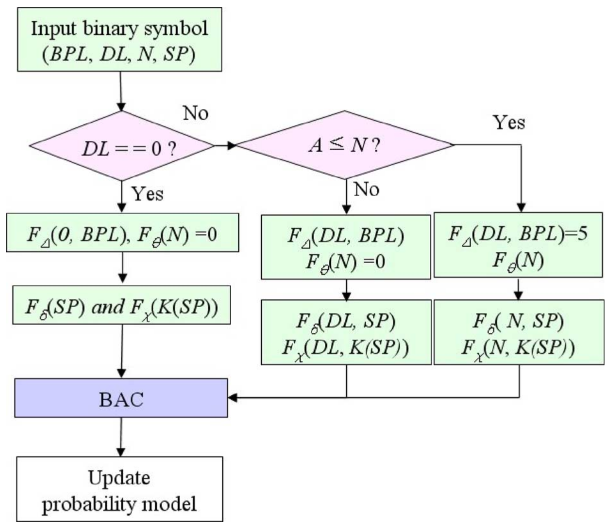

Fig. 9. Basic procedure of probability estimation.

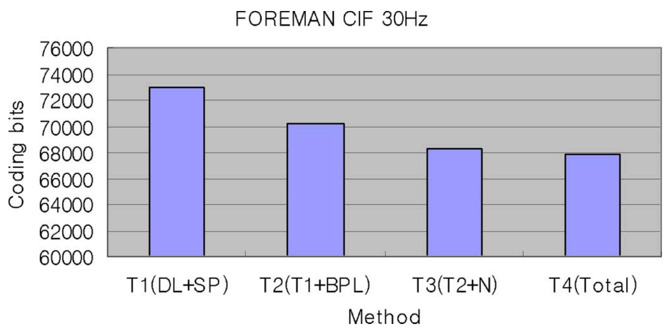

Fig. 10. Number of average coding bits according to using each proposed probability estimation method.

new interval for the symbol is repeatedly updated and calculated from the model scale of the new input symbols. In the meantime, we compare the new interval with the scale [0, T) to determine whether there is any bit for transmission down the channel. These bits are due to the redundancy in the binary representation of lower and upper values.

In Fig. 10, we compare the number of average coding bits for encoding four bit-planes in the enhancement layer according to adding each proposed probability estimation method. In T1, without considering *BPL*, the probability distribution is estimated and updated according to *DL* and *SP*. T2 and T2 additionally include *BPL* and complex-bit-plane (N), respectively. In T4, all the method described in Fig. 10 is used to estimate $\mathrm{P}(0)$. As we add each proposed probability estimation technique, we can obtain better coding efficiency.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

In our experiments, the following three kinds of FGS coding schemes have been compared using these three codecs:

1) **Codec 1:** H.264-MP4-FGS codec;
2) **Codec 2:** H.264-CBACBP;
3) **Codec 3:** JSVM [29].

We have developed **Codec 1** and **Codec 2**. In both codecs, H.264 reference software (JM9.5) has been used for base layer coding. For the FGS enhancement layer coding, MPEG-4 FGS and proposed CBACBP are implemented in **Codec 1** and **Codec 2**, respectively. In the experiments, two sets of experiments were performed. In the first experiment, the coding performance between MPEG-4 FGS and CBACBP is compared using **Codec 1** and **Codec 2**. In the second experiment, using the latest current JSVM [29] for the base layer coding, we compare the PSNR performance of all the FGS coding schemes.

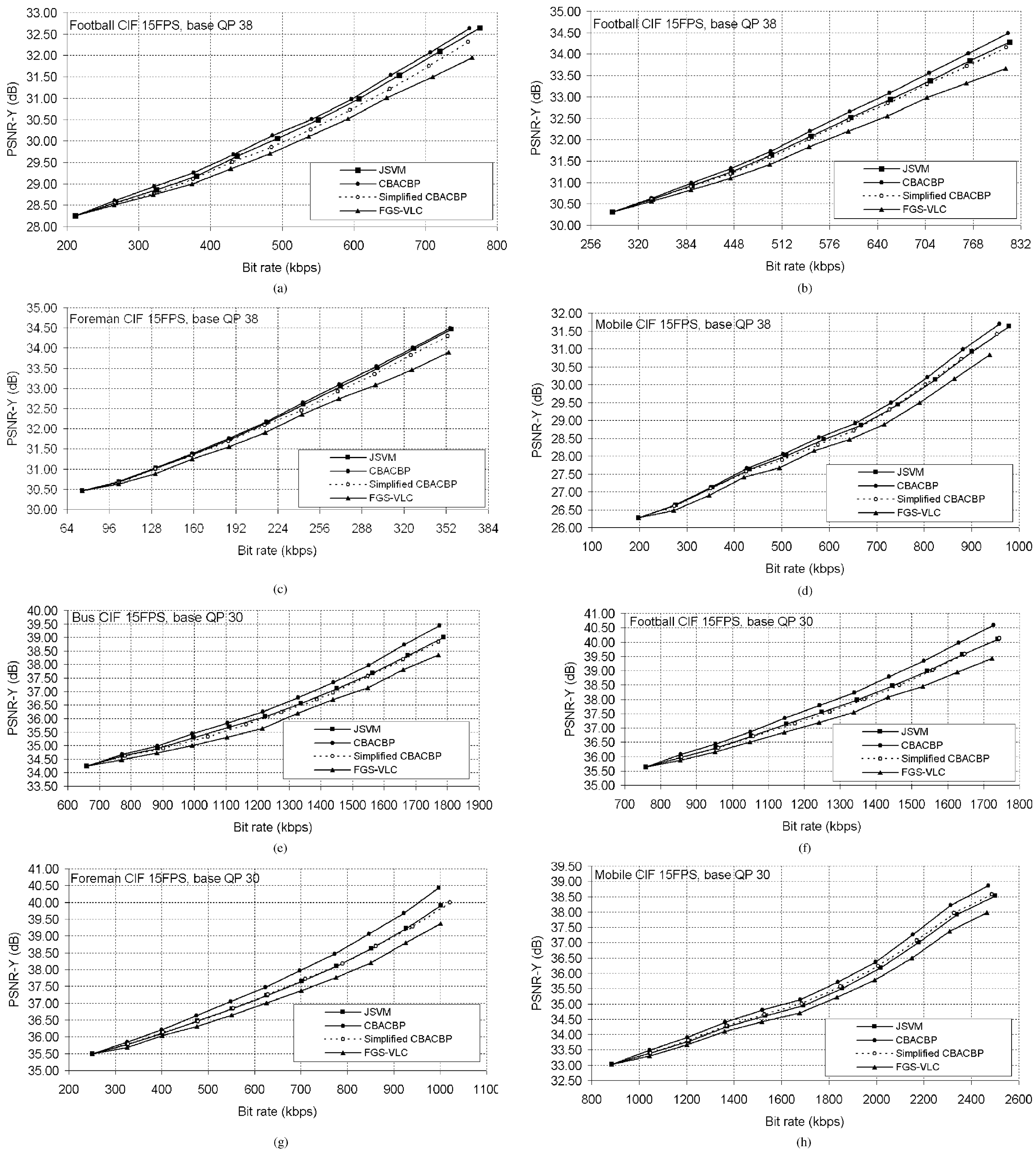

Fig. 11. Luminance PSNR comparison of FGS-VLC (MPEG-4 FGS), JSVM, simplified CBACBP, and CBACBP.

A. *MPEG-4 FGS Versus CBACBP Based on H.264*

The main goal of this test is to evaluate the coding efficiency between CBACBP and MPEG-4 FGS. In the experiments, fixed QP value (40) is used for the base layer coding. Since **Codec 1** and **Codec 2** have used the same base layer coding method (H.264: JM 9.5), the PSNR values and subjective quality of the reconstructed images are exactly the same only if the encoding of each bit-plane is finished. Therefore, in Table VI, we compare the resulting bitrate at the end-coding-points of the second and fourth bit-planes. In this experiment, we have observed that CBACBP provides bit-saving result up to 17% compared to MPEG-4 FGS.

B. *Comparison of All the FGS Coding Schemes*

In this experiment, for a fair comparison, the authors adopt an unmodified JSVM reference software with enabled AR-FGS [29]. In Fig. 11, we compare the PSNR performance of all the FGS coding schemes under the common testing conditions provided in the latest JSVM [28], [30]. In Table VII, simulation

TABLE VII
SIMULATION CONDITIONS

Sequence	BUS, FOREMAN MOBILE, FOOTBALL,
Base-layer Coding	JSVM 6.8.2 [29]
Base-layer QP	30, 38
Frame Size	CIF (352x288)
Frame Rate	15 frames/s
GOP Structure	IPPPP….

conditions are shown. For the simplified CBACBP in Fig. 11, we only consider local information (*BPL*, *DL*, and *SP*) for the context modeling. In Fig. 11, we observe that the proposed CBACBP provides better PSNR performance compared to MPEG-4 FGS (FGS-VLC) and JSVM.

V. CONCLUSION

In this paper, a new efficient entropy coding scheme, namely CBACBP, has been proposed for the enhancement layer of FGS video coding. The basic structure of proposed CBACBP is based on the traditional bit-plane coding in MPEG-4 FGS. However, in order to enhance the coding efficiency of the traditional bit-plane coding, newly designed context-based binary arithmetic coding scheme is used. In CBACBP, various context models are designed to take advantage of the characteristics and correlations of symbols in each bit-plane and different bit-planes. Experimental results show that the proposed CBACBP provides higher PSNR performance up to 0.5 and 1.2 dB than MPEG-4 FGS and JSVM FGS, respectively.

This work proves that the bit-plane coding in MPEG-4 FGS and JSVM FGS coding schemes can be further improved in coding efficiency by combining the structure of MPEG-4 FGS and a well-developed context modeling technique for the coding structure. In our research, we replace the FGS entropy coding scheme and it is an applicable solution for the low-delay application [27]. However, In order to provide the more generalized application for the variable size of GOP, the proposed scheme needed to be combined with hierarchical prediction schemes [26], [29] used in the current scalable video coding standard.

REFERENCES

[1] H. Radah, M. Van der Schaar, and Y. Chen, "The MPEG-4 fine grained scalable video coding method for multimedia streaming over IP," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, Mar. 2001.

[2] H. Radha and Y. Chen, "Fine-granular-scalable video for packet networks," *Packet Video*, Apr. 1999.

[3] Y. Chen, C. Dufour, H. Radha, R. A. Cohen, and M. Buteau, "Request for fine granular video scalability for media streaming applications," presented at the Contrib. 44th MPEG Meeting, Dublin, Ireland, 1998.

[4] W. Li, "Bit-plane coding of DCT coefficients for fine granularity scalability," presented at the Contrib. 45th MPEG Meeting, Atlantic City, NJ, Oct. 1998.

[5] F. Ling and X. Chen, "Report on fine granularity scalability using bit-plane coding," presented at the Contrib. 46th MPEG Meeting, Rome, Italy, 1998.

[6] M. van der Schaar, Y. Chen, and H. Radha, "Embedded DCT and wavelet methods for fine granular scalable video: Analysis and comparison," *Proc. SPIE IVCP*, vol. 2974, pp. 643–653, Jan. 2000.

[7] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

[8] *Information Technology–Coding of Audiovisual Objects–Part 10: Advanced Video Coding*, 14496-10:2003, 2003.

[9] J. Ascenso and F. Pereira, "H.264/AVC fine grain scalability using bit-plane coding," in *Proc. IWIAMIS*, 2004, pp. 2259–2262.

[10] J. Had, X. Sun, F. Wu, S. Li, and Z. Lu, "Variable block-size transform and entropy coding at the enhancement layer of FGS," in *Proc. IEEE ICIP*, 2004, pp. 481–484.

[11] W. H. Peng, T. Chiang, H. M. Hang, and C. Y. Lee, "A context adaptive bit-plane codec with maximum likely hood based stochastic bit reshuffling (SBR) technique for scalable video coding," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 654–667, Aug. 2006.

[12] "Scalable video coding – working draft 1," Joint video team of ITU-T VCEG and ISO/IEC MPEG, Doc. JVT-N020, 2005.

[13] , "Joint scalable video model JSVM0," Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, Doc. JVT-N021, 2005.

[14] H. Schwarz *et al.*, "Technical description of the HHI proposal for SVC CE1," ISO/IEC JTC1/WG11, Spain, Doc. m11244, 2004.

[15] "Advanced video coding for generic audiovisual services," ITU-T Rec. & ISO/IEC, 14496-10 AVC, 2005.

[16] J. Reichel, H. Schwarz, and M. Wien, "Scalable video coding – Working draft 1," Joint Video Team, Hong Kong, China, Doc. JVT-N020, 2005.

[17] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[18] "Scalable video coding – Working draft 2," Busan, Korea, JVT-O201, 2005.

[19] J. Reichel, H. Schwarz, and M. Wien, "Joint scalable video model (JSVM) 2," JVT, Busan, Korea, Doc. JVT-O202, 2005.

[20] K. Suehring, "H.264/AVC reference software," JM 9.5 [Online]. Available: http://bs.hhi.de/~suehring/tml/

[21] Y. He, F. Wu, S. Li, Y. Zhong, and S. Yang, "H.26L-based fine granularity scalable video coding," presented at the ISCAS Scottsdale, AR, 2002.

[22] J. Ridge, Y. Bao, M. Karczewicz, and X. Wang, "Cyclical block coding for FGS," ISO/IEC JTC1/SC29/WG11, M11509, 2005.

[23] J. Ridge, Y. Bao, M. Karczewicz, and X. Wang, "FGS block enhancements for scalable video coding," ISO/IEC JTC1/SC29/WG11, M11509, 2004.

[24] H. Schwarz, D. Marpe, and T. Wiegand, "MCTF and scalability extension of H.264/AVC," presented at the PCS, San Francisco, CA, 2004.

[25] H. Schwarz, D. Marpe, T. Schierl, and T. Wiegand, "Combined scalability support for the scalable extension of H.264/SVC," presented at the ICME, Amsterdam, The Netherlands, 2005.

[26] H. Schwarz, D. Marpe, and T. Wiegand, "Hierarchical B pictures," Joint Video Team, Poznan, Poland, Doc. JVT-P014, 2005.

[27] Y. Bao, M. Karczewicz, J. Ridge, and X. Wang, "Improvements to fine granular scalablility for low-delay applications," JVT, JVT-O054, 2005.

[28] Y. Bao and M. Karczewicz, "CE7 report, FGS coding for low-delay applications," JVT, JVT-Q039, 2005.

[29] "Joint scalable video model 6.8.2," ITU-T and ISO/IEC (JVT), 2006.

[30] S. Kamp and M. Wine, "Improved adaptation and coding of leak factor in AR-FGS," JVT, JVT-T062, 2006.

Seung-Hwan Kim (S'06) received the B.S. degree in electronic engineering from Ajou University, Suwon, Korea, in 2001, and the M.S. degree in information and communications engineering from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, in 2003, where he is currently pursuing the Ph.D. degree in information and communications.

His research interests include digital image and video coding, scalable video coding, multiview video coding, and digital video broadcasting.

Yo-Sung Ho (SM'06) received the B.S. and M.S. degrees in electronic engineering from Seoul National University, Seoul, Korea, in 1981 and 1983, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1990.

Since 1995, he has been with Gwangju Institute of Science and Technology (GIST), where he is currently a Professor with the Information and Communications Department. He joined Electronics and Telecommunications Research Institute (ETRI), Daejon, Korea, in 1983. From 1990 to 1993, he was with Philips Laboratories, Briarcliff Manor, NY, where he was involved in the development of the advanced digital high-definition television (AD-HDTV) system. In 1993, he rejoined the technical staff of ETRI and was involved in the development of the Korean DBS digital television and high-definition television systems. His research interests include digital image and video coding, image analysis and image restoration, advanced coding techniques, digital video and audio broadcasting, and content-based signal representation and processing.