JPEG2000 High-Speed SNR Progressive Decoding Scheme

Takahiko Masuzaki[†], Hiroshi Tsutsui[††], Quang Minh Vu[†††], Takao Onoye[††], Yukihiro Nakamura[††††]
[†]Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University
Yoshida-hon-machi, Sakyo, Kyoto 606-8501 Japan

[††] Graduate School of Information Science and Technology, Osaka University
1-5 Yamada-oka, Suita-shi, Osaka 565-0871, Japan

[†††] National Institute of Informatics, Japan
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

[††††] Research Organization of Science and Engineering, Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577, Japan

Summary
An efficient scheme for JPEG2000 SNR progressive decoding is proposed, which is capable of handling JPEG2000 compressed image data with SNR progressiveness. In order to avoid entropy decoding of the same compressed data more than once when decoding SNR progressive images, two techniques are introduced in our decoding scheme; reuse of intermediate decoding result and differential inverse discrete wavelet transform (differential IDWT). Comprehensive evaluation of our scheme demonstrating that with 26.6% increase of required memory size, up to 50% of computational cost of entropy decoding can be reduced in comparison with conventional non-progressive decoding scheme when 9/7 irreversible DWT filter is used.

Key words:
JPEG2000, Progressive decoding, Discrete wavelet transform

1. Introduction

With the rapid progress of networking technology, required functionalities for image utilization in the networked environment have changed drastically. Progressiveness of image coding, which enables us to transmit/receive image data in incremental step, has become an important factor. An example of progressive image coding is illustrated in Fig. 1, as we receive more compressed data, decoded image quality will be improved in terms of pixel fidelity, resolution, etc.

As a still image format, JPEG has been used widely such as images on the Internet and pictures by digital still cameras. However, in practice no progressiveness can be offered by JPEG. Hence there are strong demands for new image coding system with high-level progressiveness support, which can replace the position of JPEG. JPEG2000 [1], a standard of image coding system, offers many aspects of scalable coding, with which five types of progressive image streams can be produced; i.e. SNR progressive, resolution-SNR progressive, resolution-position progressive, position-component progressive, and component-position progressive. Among them, SNR progressive shows remarkable characteristics rather than other progressive modes since the mode newly introduces the notion of "layer" [2].

Fig. 1: Example of SNR progressive decoding.

To decode JPEG2000 compressed data, we need to calculate status of coefficients by classification tests, called coefficient bit modeling. This classification tests require some information obtained from the intermediate result of decoding. Thus in the decoding flow with SNR progressiveness, the decoded result for former layers is needed to decode the succeeding layer.

In conventional straightforward progressive decoding scheme, once receiving a compressed data, it is appended to the previously received data and the image is decoded using the concatenated data. However, this scheme suffers from high computational cost since the previously received data is processed many times.

Manuscript received January 5, 2009
Manuscript revised January 20, 2009

Motivated by this, in order to avoid decoding the same compressed data more than once when decoding SNR progressive images, two techniques are introduced in our decoding scheme; reuse of intermediate decoding result and differential IDWT. As for literatures on JPEG2000 progressive decoding, an unequal error protection scheme and a method that regions of interest are more quickly decoded than backgrounds are proposed in [3] and [4], respectively. These papers focus on the progressive decoding in terms of its functionality. On the other hand, this paper focuses on the system organization and its effectiveness.

This paper is organized as follows. In Section 2, we briefly introduce the encoding and decoding flow of JPEG2000 image coding system. In Section 3, we describe progressive decoding, which is one of the distinctive JPEG2000's functionalities, and conventional progressive decoding scheme. In Section 4, we propose a high-speed SNR progressive decoding scheme by introducing two efficient techniques. In Section 5, we evaluate our scheme, and the paper is concluded in Section 6.

2. JPEG2000

In this section, we briefly describe the encoding and decoding flow of JPEG2000. Fig. 2 depicts the encoding flow of JPEG2000. First, a target image is divided into square regions, called tiles. Tiles of each color component are called tile components. Then 2-D forward DWT decomposes a tile component into LL, HL, LH, and HH subbands by applying 1-D forward DWT to a tile component vertically and horizontally. The low resolution version of the original tile component, i.e. LL subband, is to be decomposed into four subbands recursively. DWT coefficients in a subband are quantized, and then the subband is divided into code-blocks, each of which is coded individually by entropy coder.

Entropy coder adopted in JPEG2000 is a context-based adaptive binary arithmetic coder, which consists of coefficient bit modeling process to generate contexts and arithmetic coding process, known as MQ-coder, to compress a binary sequence based on the context of each bit. Quantized coefficients in a code-block are separated to sign bits and absolute values, and so-called bit-planes are generated from the bits of absolute values such that each bit-plane refers to all the bits of the same magnitude in all coefficients of the code-block. Then bit-planes are coded from the most significant one to the least significant one. In the coefficient bit modeling process, a context is generated for each bit of a bit-plane based on the statistical information through three different coding passes, i.e. significance propagation pass, magnitude refinement pass, and cleanup pass, which provides efficient compression by succeeding MQ-coder.

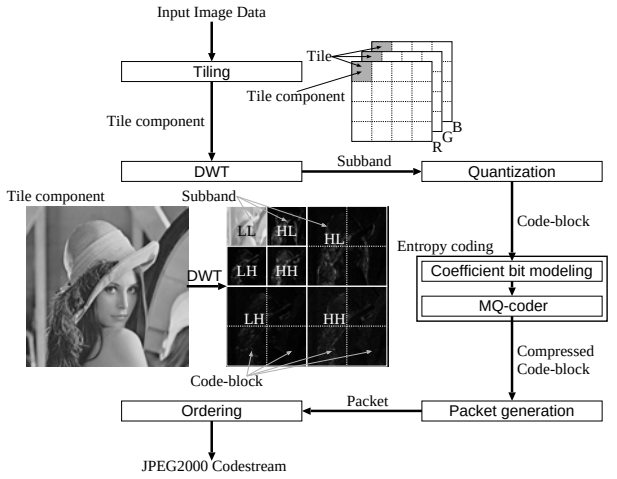

Fig. 2: JPEG2000 encoding flow.

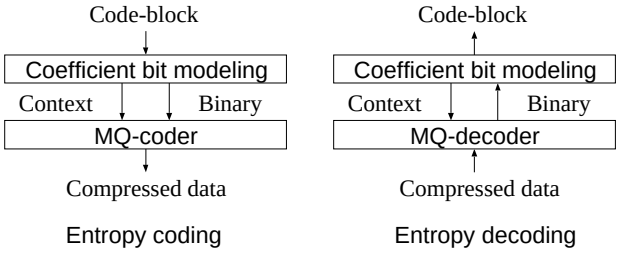

Fig. 3: Entropy coding and decoding.

As is employed in conventional DCT-base algorithms, rate control of JPEG2000 is at first roughly done by quantization process. Then, fine rate control is achieved by selecting coding pass inclusion. JPEG2000 allows selecting the number of included coding passes in a code-block arbitrary. Therefore, the rate control can be regarded as determining truncation point of MQ-coded data for each code-block, which attains better image quality. Each layer consists of such coding passes. By gathering truncated MQ-coded data originated from a specific layer, position, resolution level, and component, a packet is generated. Then re-ordering of the set of packets is applied to realize five progression modes, which is the key feature of JPEG2000 we focus on. Finally, the packets are packed into a codestream.

The decoding flow of JPEG2000 is the reverse process of the encoding. Fig. 3 depicts the flow of entropy coding and decoding. Coefficient bit modeling is a common process to encoding and decoding. MQ-decoder extracts binary sequences from compressed data referring to contexts generated by the coefficient bit modeling process using previously decoded information, while MQ-coder generates compressed data from contexts and binary sequences.

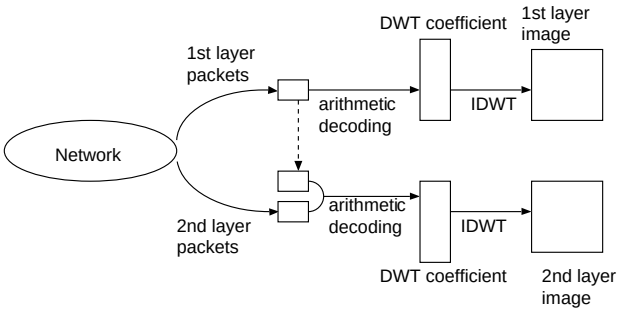

Fig. 4: Decoding flow with conventional scheme.

3. SNR Progressive Decoding

3.1 SNR Progressive Decoding

Since JPEG2000 composes codestream hierarchically, transmission of encoded image data can be achieved more efficiently compared to JPEG. Here, we consider SNR progressive to exemplify the effectiveness of the hierarchical structure. L-R-C-P progression order in JPEG2000, which implements SNR progressive, has the following nested packet structure.

for each l = 0,..,L-1
  for each r = 0,..,R-1
    for each c = 0,..,C-1
      for each p = 0,..,P-1
        packet data for component c, resolution level r, layer l, and position p.

where, L, R, C, and P denote the number of layers in corresponding tile, the number of decomposition levels, the number of color components, and the number of positions, respectively. Layers give data classification among code-blocks, each of which gathers data of code-blocks having similar contribution to image quality. When we reconstruct an image with L-R-C-P progression order, the image can be decoded as follows. Assume that only a part of codestream can be received in the beginning. With L-R-C-P progression order, we receive data sequentially from upper layer to lower layer in every code-block. Hence, even if only the data for limited layers in the code-block can be received, an image is to be reconstructed from the data received until then. Then, if additional layers are received, the decoded image reflects more faithfully to the original one by using the additional data as well as the data received in advance. As much layers we receive, the more image quality can be achieved owing to the layer structure. In other words, receiving codestream in incremental step, the image quality in terms of signal-to-noise ratio is raised incrementally.

3.2 Conventional Progressive Decoding Scheme

In order to decode JPEG2000 codestream, we need to calculate status of coefficients, called “context label”. Context label is determined by classification tests such as whether a coefficient is significant or not, whether the coefficient is positive or negative, and whether the coefficient has decoded in magnitude refinement pass. Information needed for the classification tests can be obtained from the intermediary decoded data. Thus in the decoding of codestream with SNR progressiveness, the decoded result for former layers is needed to decode the succeeding layer.

In conventional progressive decoding scheme, first all the codestream data needed for decoding must be stored into memory, and then start image decoding. Hence the following serious overhead may be brought about in the incremental step decoding stated above. Once receiving a partial codestream, the scheme stores it to the memory and decodes to reconstruct low quality version of an image. If additional codestream data is received, the scheme locates the data to the next to the codestream data received in advance as illustrated in Fig.4. Then the scheme starts again the decoding from the beginning of the codestream data so as to attain higher quality version of the image.

This tedious decoding is due to the fact that arithmetic decoding of the newly received data is highly dependent on the result of the previously received codestream, and the decoding cannot be started from the mid-flow. As a result, with this non-progressive decoding scheme we need to execute entropy decoding for the same data more than once. Considering that the arithmetic decoding is the most computationally intensive among whole decoding processes, which occupies about 60% of all calculations [5], this non-progressive decoding scheme can be hardly regarded as a practical one in terms of computational costs.

4. High-Speed SNR Progressive Decoding

To resolve the decoding inefficiency described above, high-speed progressive decoding scheme is proposed in this paper, which enables us to resume decoding at the mid-flow of JPEG2000 stream as illustrated in Fig. 5.

4.1 Reuse of Intermediate Decoding Result

As mentioned in Section 2, in order to gain the coding efficiency in arithmetic coding, JPEG2000 puts the same context label to coefficients having similar probability, and encodes all coefficients in three passes, i.e. significance propagation pass, magnitude refinement pass, and cleanup pass, with updating probability

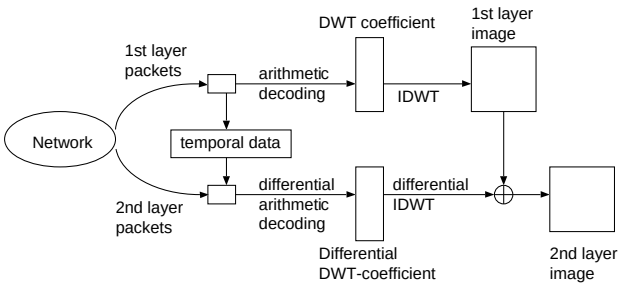

Fig. 5: Decoding flow with proposal scheme

adaptively. Context label is determined by classification tests as described in the following.

In the process of encoding, all values of coefficients are known in advance so that context labels can be easily determined. On the other hand, in the process of decoding, context labels must be determined by using the previously decoded data. Thus, information needed for the classification tests must be stored as a temporal result. In addition to this, information about in which pass a coefficient in a bit-plane is included is also needed. The pass is basically decided from the information whether the coefficient in the bit-plane is significant or insignificant. However, as for magnitude refinement pass and cleanup pass, additional information is still needed. In magnitude refinement pass, significant coefficients are decoded. However, if the coefficient becomes significant for the first time in significance propagation pass, this coefficient is treated as significant in magnitude refinement pass. Therefore, we need to determine whether to decode a coefficient in magnitude refinement pass by referring to information about coefficients which have already been decoded in significance propagation pass.

Consequently, the temporal information to be kept is the result of following four classification tests:

1. whether the coefficient is significant or insignificant,
2. whether the coefficient is positive or negative,
3. whether the coefficient has already been decoded in magnitude refinement pass, and
4. whether the coefficient has already been decoded in significant propagation pass.

A trivial way to keep these 4 values is to assign 1 bit for each. However, only 8 patterns out of 16 patterns are occurred in practice. Therefore, we can save the temporal information with 3 bits per DWT coefficient.

4.2 Differential Inverse Discrete Wavelet Transform

Output from entropy decoder is processed with IDWT. However, using proposed intermediate result reusing scheme, entropy decoder outputs only the processed bit-plane, i.e. 1 bit of DWT coefficient, instead of whole DWT coefficient. Therefore, each DWT coefficient is necessarily saved to apply IDWT in addition to the decoded image. Since saved DWT coefficients need to be updated by using newly decoded bit-planes, we must save all DWT coefficients of the image. In order to avoid this overhead, we newly introduce the technique, which allows us to calculate IDWT by using only one bit-plane.

The original 1D-IDWT is calculated by using following equations.

5/3 Filter (reversible compression)

IDWT for reversible compression can be executed by

$$X(2n) = Y(2n) - \left\lfloor \frac{Y(2n-1)+Y(2n+1)+2}{4} \right\rfloor,$$

$$X(2n+1) = Y(2n+1) + \left\lfloor \frac{X(2n)+X(2n+2)}{2} \right\rfloor,$$

where, $Y(n)$ and $X(n)$ are input and output sequences of coefficients, respectively.

9/7 Filter (irreversible compression)

IDWT for irreversible compression is executed by a series of STEP1&2, STEP3&4, and STEP5&6.

$$\begin{aligned}
STEP1: &\quad X(2n) = KY(2n) \\
STEP2: &\quad X(2n+1) = \frac{1}{K}Y(2n+1) \\
STEP3: &\quad X(2n) = X(2n) - \delta(X(2n-1)+X(2n+1)) \\
STEP4: &\quad X(2n+1) = X(2n+1) - \gamma(X(2n)+X(2n+2)) \\
STEP5: &\quad X(2n) = X(2n) - \beta(X(2n-1)+X(2n+1)) \\
STEP6: &\quad X(2n+1) = X(2n+1) - \alpha(X(2n)+X(2n+2))
\end{aligned}$$

where, α, β, γ, δ and K are constant values for filter processing.

Henceforth, the flow of differential IDWT is described. This technique is based on the fact that decoding of a new bit-plane is equivalent to get the difference of DWT coefficient. This is reasonable since there is the linearity in the equations of original IDWT except for the floor operation in reversible compression. Thus as for the irreversible compression, we just need to calculate the differential value for a new bit-plane. As for the reversible compression, the differential IDWT can be achieved by holding the residue of floor operation for each coefficient.

In concrete terms, the processing flow is as follows.

5/3 Filter

$$\Delta X(2n) = \Delta Y(2n) - \left\lfloor \frac{\Delta Y(2n-1)+\Delta Y(2n+1)+2+r_{2n}}{4} \right\rfloor,$$

$$\Delta X(2n+1) = \Delta Y(2n+1) + \left\lfloor \frac{\Delta X(2n)+\Delta X(2n+2)+r_{2n+1}}{2} \right\rfloor,$$

$$r'_{2n} = (\Delta Y(2n-1) + \Delta Y(2n+1) + 2 + r_{2n}) \mod 4,$$
$$r'_{2n+1} = (\Delta X(2n) + \Delta X(2n+2) + r_{2n+1}) \mod 2,$$

where r_n and r'_n are current and next residues to be kept.

9/7 Filter

$$\begin{aligned}
STEP1: \quad \Delta X(2n) &= K\Delta Y(2n) \\
STEP2: \quad \Delta X(2n+1) &= \frac{1}{K}\Delta Y(2n+1) \\
STEP3: \quad \Delta X(2n) &= \Delta X(2n) - \delta(\Delta X(2n-1) + \Delta X(2n+1)) \\
STEP4: \quad \Delta X(2n+1) &= \Delta X(2n+1) - \gamma(\Delta X(2n) + \Delta X(2n+2)) \\
STEP5: \quad \Delta X(2n) &= \Delta X(2n) - \beta(\Delta X(2n-1) + \Delta X(2n+1)) \\
STEP6: \quad \Delta X(2n+1) &= \Delta X(2n+1) - \alpha(\Delta X(2n) + \Delta X(2n+2))
\end{aligned}$$

Since real number operations are executed in the irreversible compression, the result of the proposed scheme may differ from the result of the original DWT. However, owing to the linearity of equations for both proposed and original DWT, the error can be suppressed to negligible value.

A result of the differential IDWT is shown in Fig.6.

Fig. 6(a): 1st layer image

Fig.6(b): differential image (biased)

Fig. 6(c): image from all streams

5. Evaluation

In this section, evaluation of the proposed high-speed SNR progressive decoding scheme is explained in terms of memory size and computational costs.

5.1 Memory Size Evaluation

For this memory size evaluation, we assume that the input image size is $w \times h$ pixels, comprising 128×128 sized-tiles, and that each pixel is grayscale and 8bit depth. Memory for image decoding is used mainly for I/O buffer in IDWT and stream buffer to store I/O stream. We also assume that the decoder processes tiles one by one.

In the calculation of DWT, the number of output bits increases from the number of input bits so as to avoid overflow occurrence. In practice, this increased number differs among DWT subbands, however to make explanation simple all DWT inputs and outputs are assumed to 8bits and 12bits, respectively. This means that the IDWT inputs and outputs are 12bits and 8bits, respectively. In this case, I/O buffer size for IDWT is $8 \times wh + 12 \times 128 \times 128$ bits. Also memory for JPEG2000 stream buffer becomes $8cwh$ bits, when the worst-case compression ratio is c. Consequently, total memory size needed for decoding with conventional scheme is $8(1+c)wh + 12 \times 128 \times 128$ bits. Note that the memory for input of IDWT need to store DWT coefficients for only one tile, since we are assuming that the decoder processes tiles one by one. If differential IDWT is not used, we need to store all of the previously decoded DWT coefficients as mentioned in the previous section, and the memory size would be $8(1+c)wh + 12wh$ bits.

For decoding by the proposed scheme, we need temporal data memory along with the memory stated above. First, we need to store the coefficient's state to determine the context label and decide a pass to decode the coefficient. There are eight states of coefficient, thus 3bits per coefficient is needed to store the state. Next, in the case of 5/3 filter, we need to store the residues of floor operation to calculate differential IDWT. In that floor operation we divide coefficients by 4 or 2, and the range of residue value is 0~3 or 0~1 for which 1.5 bits is needed for each coefficient in average. Since this 1.5 bits is needed for each 5/3 filter operation, the total amount of bits is $1.5 \times 2 \times wh \times (1+1/4) = 3.75wh$ considering 2D differential IDWT and recursive IDWT in case the number of DWT decomposition is 2. Summing up these two temporal data, the needed temporal data size is $6.75wh$ bits. Besides this temporal data, we need a memory to keep the state of entropy decoder. In order to hold this temporal data for MQ-decoder, 25bytes = 200bits are needed per code-block. Then assume that the size of code-block is 32×32 and that the number of DWT decomposition is 2, the number of code block is $wh/(32 \times 32)$. Hence, the temporal data from entropy decoder is $wh/(32 \times 32) \times 200$ bits. Consequently, the increasing memory size to use proposed scheme is $6.75wh + wh/(32 \times 32) \times 200 = 6.95wh$ bits. The ratio of memory increase for the proposed scheme to the conventional non-progressive decoding scheme is

$$\frac{6.95wh}{8(1+c)wh + 12 \times 128 \times 128},$$

which is about 57.9% when c=1/2 and $w \times h$ is large enough. On the other hand, in the case of 9/7 filter, we need not to store the residue of floor operation, therefore the ratio of memory increase is

$$\frac{3.19wh}{8(1+c)wh + 12 \times 128 \times 128},$$

which is about 26.6% in the same condition.

5.2 Computational Costs Evaluation

Then, we evaluate the computational costs for entropy decoding. In this evaluation, we assume that the computational cost of entropy decoder is proportional to the amount of input data to the entropy decoder.

There is a trend in progressive image coding that as the image quality increases, the data size required for more improvement of the image quality increases. Therefore, codestream is usually divided into layers as follows. First, assign half of all the codestream into the first layer. Next, divide the rest of codestream into two, and assign the former piece into the 2nd layer. In this manner, all codestream data are assigned to layers. For example, in the case of dividing the codestream into three layers, layer 0 and layer 1 holds 1/4 of the entire stream, and layer 2 holds the rest 1/2 of the entire stream.

If we decode this codestream with the conventional scheme, calculations are to be done as follows. Decoding of layer 0 requires to process 1/4 of all image data. Decoding from layer 0 to layer 1 requires to process 1/4 + 1/4 = 1/2 of all image data. Then, decoding from layer 0 to layer 2 requires to process 1/4 + 1/4 + 1/2 = 1/1 of all image data. Thus decoding all of these images, each of which corresponds to a quality level given by a specific layer, requires to process totally 1/4 + 1/2 + 1/1 = 7/4 of the original image data.

In contrast, the computational costs with the proposed high-speed SNR progressive decoding are as follows. In this scheme intermediate decoding result is stored, hence if there are multiple layers, we can start decoding from mid-flow of image stream. Thus there is no need to decode the same data more than once. Therefore the calculation amount with proposed scheme is proportional to the image data size.

As a result, comparing to the conventional scheme, we can reduce (7/4−1)/(7/4) = 42.9% of computational costs in this example. As the number of layers increases, computational costs with the proposed scheme approaches to 50%.

6. Conclusion

In this paper, we proposed JPEG2000 high-speed SNR progressive decoding scheme. With the proposed scheme, we can resume decoding from the mid-flow of JPEG2000 codestream, instead of decoding from the start of codestream. We proposed reusing scheme for intermediary decoded data with holding the temporal information needed to resume decoding from the mid-flow.

In this SNR progressive decoding scheme, arithmetic decoder outputs only one bit-plane of DWT coefficients. Comparing to the conventional IDWT, which needs to gather all bit-planes of IDWT coefficient, the proposed scheme is capable of applying differential IDWT using only one bit-plane of DWT coefficient.

In the case of 9/7 irreversible filter, by increasing 26.6% of memory usage, the proposed scheme reduces up to 50% of computational costs, which is practical enough for receiving the benefit of JPEG2000 progressive coding.

References

[1] ISO/IEC JTC 1/SC 29/WG 1 N2678, “JPEG 2000 part 1 020719 (final publication draft),” 2002.

[2] D. Taubman, “High performance scalable image compression with EBCOT,” IEEE Trans. on Image Processing, vol.9, no.7, pp.1158−1170, Jul. 2000.

[3] L. Pu, M.W.Marcellin, B.Vasic, and A.Bilgin, “Unequal error protection and progressive decoding for JPEG2000,” Image Communication, vol.22, no.3, pp.340−346, Mar. 2005.

[4] O. Watanabe, and H. Kiya,！“An extension of ROI-based scalability for progressive transmission in JPEG2000 coding,” IEICE Trans on Fundamentals, vol.E86-A, no.4, pp.765−771, Apr. 2003.

[5] H. Tsutsui, T. Masuzaki, Y. Hayashi, Y. Taki, T. Izumi, T. Onoye, and Y. Nakamura, “Scalable Design Framework for JPEG2000 System Architecture,” in Proc. ACSAC2004, pp.296−308, Sep. 2004.

Takahiko Masuzaki received B.E. and M.S. degrees from Kyoto University, Japan, in 2001 and 2003, respectively. He is currently a researcher in Mitsubishi Electric Corp. His research interests include LSI design methodology. He is a member of IEEE, IEICE, IPSJ, and ITE-J.

Hiroshi Tsutsui received B.E., M.S., and Ph.D. degrees from Kyoto University, Japan, in 2000, 2002, and 2005, respectively. He is currently an assistant professor in the Department of Information Systems Engineering, Osaka University. His research interests include image processing and its implementation for embedded systems. He is a member of IEEE, ACM, IEICE, IEEJ, and IIEEJ.

Quang Minh Vu received his B.E. from Kyoto University in 2003, his M.E. from the University of Tokyo in 2005 and his Ph.D from the University of Tokyo in 2008. He has done researches in several research areas in computer science: computer hardware optimization, network architecture design, and text data mining. He is now a postdoctoral researcher at National Institute of Informatics, Japan, doing a research on a management system of bibliography database. He is a member of IPSJ.

Takao Onoye received B.E. and M.E. degrees in Electronic Engineering, and Dr.Eng. degree in Information Systems Engineering all from Osaka University, Japan, in 1991, 1993, and 1997, respectively. He is currently a professor in the Department of Information Systems Engineering, Osaka University. His research interests include media-centric low-power architecture and its SoC implementation. He is a member of IEICE, IEEE, IPSJ, and ITE-J.

Yukihiro Nakamura received his B.S., M.S., and Ph.D. degrees in Applied Mathematics and Physics from Kyoto University, in 1967, 1969 and 1995, respectively. From 1969 to 1996, he was with Electrical Communications Laboratories, NTT. In NTT he engaged in research and development of the behavioral description language ``SFL'' and the High-Level Synthesis System “PARTHENON”. Concurrently, he was a guest professor at Graduate School of Information Systems, University of Electro-Communications. In 1996, he joined Graduate School of Informatics, Kyoto University as a professor. Since 2007, he has been a professor of Research Organization of Science and Engineering, Ritsumeikan University. He has also been a coordinator of Synthesis Corporation since 1998. He received Best Paper Award of IPSJ, Okochi Memorial Technology Prize, Minister's Prize of the Science and Technology Agency and Achievement Award of IEICE in 1990, 1992, 1994, and 2000, respectively. He is a fellow of IEEE and a member of IEICE, IPSJ, and ACM.