An Adaptive Block Truncation Coding Scheme and Its Data-Driven Parallel Implementation

Xiaoyan Yu

A dissertation submitted to
Kochi University of Technology
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

(Special Course for International Students)
Department of Engineering
Graduate School of Engineering
Kochi University of Technology
Kochi, Japan

March, 2006

Abstract

An Adaptive Block Truncation Coding Scheme and Its Data-Driven Parallel Implementation

Xiaoyan Yu

In this dissertation, an adaptive block truncation coding (ABTC) scheme suitable for highly-parallel software implementation is proposed. ABTC is one of extended schemes of the original block truncation coding (BTC), but it introduces a new classification approach for adaptive coding and differential pulse coding modulation (DPCM). Furthermore, its highly-parallel data-driven software implementation is discussed to realize real-time video applications on small and low-power ubiquitous devices.

In the proposed ABTC scheme, two optimal threshold values are introduced to identify a luminance block image as uniform block, normal block, or pattern block. One is the sample first absolute central moment (AM), which denotes the dispersion from the mean value in a 4x4 pixel block; another is the mean of absolute errors (MAE) between the original pixel values and their decoded data in every block image, which is computed with the simplified absolute moment block truncation coding (SAMBTC). In order to achieve a better trade-off between the image quality and computational complexity, different coding approaches are employed to compress/decompress three sorts of block images. Moreover, to improve the compression efficiency, DPCM is utilized to remove the redundancies existing in the intra- and inter-frame by variant prediction methods with the negligible image distortion.

In the data-driven parallel implementation, the following techniques are utilized to achieve a high-throughput performance. First, hierarchical parallelism inherent in the ABTC scheme is exploited to utilize the hardware resources at maximum. Second, the computational complexity is decreased by taking full advantage of the compound operators involved in the current instruction sets. Also, associative temporal memories are used to realize the distributed computation. Third, the SIMD-type (single-instruction-multiple-data) packet is employed to accomplish the data-level parallelism (DLP) and to decrease the pipeline processing load. Finally, static load balance is discussed for the scalable implementation on the available processing resources such as processors and memories.

Experimental evaluations of the proposed scheme were performed using standard video sequences depicting variant amounts of motion and background activity. The results illustrate that the proposed scheme can achieve reconstructed image sequences up to around 37 dB with 60 compression ratio. Compared with previous scheme, over 1 dB image quality gains can be obtained with the identical compression ratio. Furthermore, evaluation result of DDMP implementation shows over 60 VGA frames per second on average, which is around twofold as many as that of previous scheme.

key words data-driven, adaptive BTC, DPCM, parallel implementation

Acknowledgement

First of all, I would like to appreciate Kochi University of Technology and President of KUT, Prof. Hajime OKAMURA, for giving me this chance to study here and offering three years of expense support. Without the special scholarship program (SSP) for international students, I cannot complete my doctoral research successfully.

Next, I am very thankful of my supervisor, Prof. Makoto IWATA, for three years of encouragement, guidance and support. Prof. Makoto IWATA has guided me into the exciting area of image processing and computer vision and data-driven parallel computing of high performance. I had never forgotten his saying that "Never give up" when you meet the difficulty in your research. Moreover, his research enthusiasm and insight into data-driven architectures and software system make me deep impression. Especially, his patience and serious spirit for study and instructing students will affect my career forever. The work presented here would not have been possible without his constant guidance and helpful recommendation. I am especially grateful for his untiring effort to teach me how to think to tackle the problems in my research world. I believe his good thinking way and research skills will be beneficial to my study in the future. I would also like to thank other members of my committee, Prof. Takao NISHITANI, Prof. Kazunori SHIMAMURA, Prof. Mamoru OKADA, and Associate Prof. Masahiro FUKUMOTO, for helpful comments on my thesis work.

Besides, I would like to appreciate the other professors and secretaries of the department of Information Systems Engineering for their help. Moreover, I acknowledge the staffs of the international relations center, Prof. Mikiko BAN, Ms. Mariko KUBO, Ms. Yuriko HAMADA, Ms. Kimi KIYOOKA, for their kindness and warmness.

Third, I am grateful of the fellow students in Iwata Lab. It would have been far more difficult to complete this work without their help. Special thanks go to Daichi Morikawa, Ruhui Zhang, Shuji Sannomiya, Hiroshi Shima, Shinji Ogasawara, Shirane Yuta, Shouji Tsuneishi for their helpful discussion and advice. I would have never forgotten their help and warmness which made my academic experience at KUT most enjoyable. Again, I would like to thank the other SSP students of KUT for their help.

Last but not least, my deepest gratitude goes to my dear husband for his deep love and persistent support. He has been my advisor, critic, teacher, co-investigator, close friend, and partner from before the beginning of this Ph.D work and through its evolution. I appreciate his sacrifices that made this research and my degree possible forever. Thank you to my mother, Xiuyun Wang, for giving me her wholehearted support through all these years. To my father, Jingjun Yu, thank you for teaching me how to make correct decision on every critical turn in my life. I am indebted to my younger sisters, Xiaodong Yu and Xiaolan Yu for their considerate care to our parents during the period I studied abroad.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

With the rapid development of networking and multimedia technology, digital video can be found in a variety of application fields. These applications include wire-line and wireless real-time conversational services, internet or local area network (LAN) video streaming using Real-Time Protocol/Internet Protocol (RTP/IP), and storage formats (e.g. digital versatile disk (DVD), digital camcorders, and personal video recorders.) [1]. In the networked digital video home (NDVH) scenario, the family members can access the IP-based TV program, video on demand, video games, and interactive services via WiFi, Bluetooth, Ethernet or Power line (Homeplug). Such kind of services can be provided by the prevalent electronics such as digital TV sets and multimedia mobile handheld devices. It can be imagined that video stream will appear in the cars soon.

In order to meet a wide variety of multimedia applications mentioned above, various video coding standards have been developed by international standard organization (ISO)/international electrotechnical commission (IEC) and international telecommunication union (ITU) over the past years. On the one hand, a family of motion picture experts group (MPEG) coding standards like MPEG-1, MPEG-2 and MPEG-4 are formed by ISO/IEC MPEG. On the other hand, a series of H.26x coding standards as H.261 and H.263 are established by ITU telecommunication standardization sector (ITU-T). Moreover, the emerging “Advanced Video Coding” standard known as ITU-

Table 1.1 Basic features of video coding standards.

	Transform MC Block Size	MC Accuracy	Additional Motion Prediction Modes	Target Applications
MPEG-1 (1993)	8x8 DCT 16x16	-pel	-B-Frames	Storage of compressed video on CD
MPEG-2 (1995)	8x8 DCT 16x16, 8x16	-pel	-B-Frames -Interlace	Digital TV signals around 4-6 Mb/s
MPEG-4 (2000)	8x8 DCT 8x8, 16x16	-pel	-B-Frames -Interlace -Global MC	Very low bit rates and providing object-based functionalities
H.261 (1993)	8x8 DCT 16x16	1-pel	-B-Frames	Digital video telephone signals over ISDN
H.263 (1995)	8x8 DCT 8x8 8x16	-pel	-B-Frames	Video-telephone signals at low bit rates
H.264 AVC (2002)	4x4 16x16, 16x8 8x8,8x4,4x4	$\frac{1}{8}$-pel	-B-Frames, -In-loop deblocking filter -CAVLC/CABAC	All application domains over a broad range of bit rates

T Recommendation H.264 and as ISO/IEC MPEG4 Part 10 is standardized by both ITU-T and ISO/IEC. The basic features of the developed coding standards are shown in Table 1.1 [2]. As illustrated in Table 1.1, 8x8 DCT is utilized in the existing standards except for H.264. To reduce the computational complexity of transform, the 4x4 integer DCT is used in H.264 standard. Furthermore, it can be seen from Table 1.1 that the latest coding standards like MPEG-4 and H.264 become more complex than the former ones such as MPEG-1/MPEG-2 and H.261/H.263 for the improvement of coding efficiency. It can also be derived that each of earlier standards is targeted at some special application field, while the latest ones own a broad range of application domains through increasing functional modules.

Even though 2-D video stream has been penetrated into so many fields, establishing a practical video communication system is still so expensive for us due to high hardware cost. Since the hardware cost of video codec system is critical to broaden video communication applications, it must be minimized. Currently, there are two alternative ways to reduce the chip count, system-on-chip (SoC) or system-in-package (SiP) solution. Both approaches have merits and accomplish the goal. SoC ultimately offers the lowest silicon cost, but SiP approach enables more flexibility, which is potential to re-use existing software, development tools across a range of models. Each of them available with the advent of deep sub-micron LSI fabrication has been a better cost-effective solution for video communication system on ubiquitous devices such as mobile phone and laptop. This is because they can reduce the wire length and system size so as to diminish the power consumption. Power saving is a critical feature for the battery-powered portable devices. Moreover, communication overhead between chips can be eliminated to achieve a higher processing power than system composed of independent component elements. It is the superior features of SoC and SiP that the current multimedia mobile devices such as smart mobile phone have been built on it. For instance, media-embedded processor (MeP) SoC developed by Toshiba [3] is targeted at digital signal processing such as video and audio.

In order to realize a flexible embedded system-on-chip (SoC) system, it is important to develop a fast video codec system which flexibly operates on a programmable SoC system. While emerging coding standards shown in Table 1.1 endure heavy computational complexity, it is hard to realize a flexible real-time coding on SoC by software solutions. Consequently, to develop a simple and fast video codec system without complex functions such as the motion estimation and compensation, many application specific IC (ASIC) chips and intellectual property (IP) LSI building blocks dedicated to an image codec such as Motion-JPEG [4] and Motion-JPEG2000 [5] have been developed. However, their operating functions cannot be changed flexibly because they are hardwired. Even though today's CPU speed has been improved greatly by increasing clock rate, to establish a powerful software implementation platform which can empower the emerging complex coding standards to realize a real-time video coding on the flexible SoC is still not easy. This is because in this way the system intends to suffer from the clock skew problem and incremental power consumption associated with the synchronous clock distribution. Consequently, it it more difficult to continuously increment CPU speed by taking advantage of increased clock rate.

Alternatively, a video coding scheme and its software parallel implementation are considered together to design a video codec system on the flexible SoC. When designing the video codec system, both simplification and parallelization of the video coding scheme are performed simultaneously in order to reach a desired throughput performance for embedded video applications. Using algorithm/implementation co-design approach for developing SoC system, the design period is reduced too much compared with the algorithm design independent from its implementation. It can be therefore derived that this co-design method is more cost-effective than the independent design for the coming multi-functional and flexible SoC.

1.2 Modern Video Coding Algorithms

Since a stream video involves a huge amount size of entropy information, it is important to minimize the size of digital video to custom storage device or networking bandwidth while keeping the best image quality. In order to decrease the adoption size

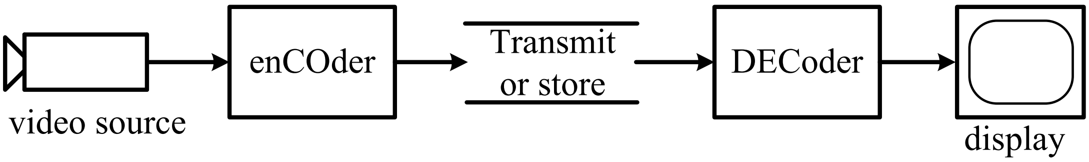


Fig. 1.1 Video codec block diagram.

of digital video across a wide range of embedded applications, a video codec system shown in Fig. 1.1 [2] is required. It can be observed that the codec system involves a complementary pair of systems, an encoder and a decoder. The encoder converts the source data into a compressed form (occupying a reduced number of bits) prior to transmission or storage and decoder converts the compressed form back into a representation of the original video data. With regard to establishment of the codec system, a coding algorithm and its implementation platform are essential. In order to achieve a satisfactory reconstructed image with the bandwidth limitation of transmission channel, an efficient coding algorithm should be robust to remove the redundant information inherent in the image and consecutive images. To this end, great efforts have been made to develop efficient coding algorithms to meet continuously emerged multimedia applications during the past decades.

Image/video coding has been studied for around half of century since 1950s and 1960s with spatial differential pulse coding modulation (DPCM) coding of images. Along with the increase in multimedia applications of consumer products, a sequence of video coding standards shown in Table 1.1 have been developed for different target applications. It can be derived that coding approaches such as spatial-frequency transform and motion compensation (MC) are continually evolved for better coding gains. Generally speaking, the primary modules of the coding standards include image transformation, motion estimation/compensation and entropy coding.

In order to remove the redundancy inherent in the image or residual image after prediction, image transformation is utilized. Image transformation is targeted to extract the low-frequency entropy and remove the high-frequency one so as to reduce the image size. This is based on the characteristics of Human Vision System (HVS). Human is more sensitive to the low frequency information than the high frequency one. Concerning image transformation, three techniques widely used in the modern coding standards are discussed below.

The first is block-based discrete cosine transform (DCT), introduced by Ahmed et al. [6] in 1974 for intra-frame coding. Since then, subsequent image and video coding standards like JPEG, MPEG1 and MPEG2 include DCT to remove the redundancies existing in the image or residual image after prediction. The second is discrete wavelet transform (DWT) technique used for image and video coding starting in the 1980s. Nowadays, DWT provides the core technology for the MPEG4 texture coding standard and JPEG2000 still image coding standard. DWT has been proved significant coding gains compared with DCT at the cost of more complexity. The third is Hadamard transform used for the newest coding standard H.264/MPEG 4 part 10. Hadamard transform is an approximation of 4x4 DCT but it uses the integral transform for simplification of computation at the cost of loss of accuracy. Moreover, the Hadamard coding intends to result in around 20% increase of the access frequency. Also, at the decoder, it increases the decoding time up to 5% [7].

Motion estimation and compensation (ME/MC) technique is widely utilized to remove the inter-frame redundancies inherent in the consecutive frames. ME/MC started in the 1970s and matured into practical technology around 1985 with the advent of the basic hybrid block-based motion compensation/DCT systems (MC/DCT). MC/DCT coding strategies are implemented in all of today's ISO/IEC MPEG and ITU video coding algorithms [8].

MC/DCT technology provides a significant compression gain versus pure intra-frame DCT coding (i.e., JPEG) for video compression. The major video coding standards released since the early 1990s such as H.261, H.263, MPEG-1, MPEG-2, MPEG-4 visual and H.264, have been based on the identical basic MC/DCT design (or model). These video coding standards implemented the similar set of basic encoding and decoding functional modules although there are many differences of details between the standards and implementations.

In order to increase the coding gains, much more complexity is added to the encoder which has to perform motion estimation and compensation (MC). Recent extensions of the basic MC/DCT approach (i.e., those standardized with H.263, MPEG-1/2/4, H.264) have further improved compression efficiency at the expense of more complex encoders and decoders.

For example, the latest H.264/AVC standard provides gains in compression efficiency of up to 50% over a wide range of bit rates compared to previous standards [7]. The superior performance of H.264 to its predecessors is attributed to a number of relatively new technical improvements adopted, such as $\frac{1}{4}$ pixel inter-frame comparison on blocks ranging from 4x4 to 16x16 pixels. Nevertheless, the complexity of H.264 decoder is about four times that of MPEG-2 decoder and two times that of MPEG-4 decoder (Visual Simple Profiles) [7].

In addition to aforementioned two techniques, the third significant task is entropy coding used to remove the statistical redundancies. Three sorts of entropy coding approaches with diverse computational complexity and coding efficiency have been exploited so far. The simplest one is variable length coding (VLC) widely utilized in the earlier MPEG and H.26x coding algorithms. The second one is context-adaptive variable length coding (CAVLC) used in the baseline and extended profile of H.264, which can produce higher coding efficiency than VLC at the cost of more complexity. The third one is context-adaptive binary arithmetic coding (CABAC) used in main file of

H.264, which reduces the bit rate up to 16% at the exchange of an access frequency increase from 25 to 30% compared to a single reversible VLC table for all syntax elements [2].

Besides the image and video coding standards mentioned above, digital signal coding techniques include the block truncation coding (BTC) [9], vector quantization (VQ) [10] and fractal image coding [11]. BTC was originally developed at Purdue University. Since BTC has the advantages including the preservation of signal pixel resolution and edges and low computational complexity, it is extensively used in many image compression applications [12]. Vector quantization [10] is another simple image coding technique. For image compression using VQ, a macro block of pixels is approximated by the maximum-likelihood pattern in the codebook, where typical patterns occurring in video images are stored as templates, and only the code of the best matched pattern is transmitted. At the decoder, the corresponding pattern in the codebook will be used to reconstruct this block image. Since the pattern matching is a very computationally expensive process, VQ is asymmetric in compression and decompression operations. Thus VQ is so far suitable only for one-way video transmission in portable applications. As for fractal image compression techniques, there are some attractive advantages such as resolution independence, fast decoding and very competitive rate-distortion curves. Despite the advances made, the long encoding time still remains the main drawback of this technique [11]. So it may not be suitable for real-time communication applications such as person-to-person teleconferencing.

From the development history of image and video coding, it can be derived that the existing coding standards focus on the decrease of bit rate and a broad range of applications. As a consequence, the higher coding efficiency of the latest coding standards like H.264 than their predecessors is obtained by adding several unpredicted elements. Nevertheless, these unpredicted elements tend to increase the implementation complexity. Also, the coding standards were proposed for the sake of generic application fields, thus the image characteristics of a specific application were omitted. Accordingly, as for a specific application, the existing coding standards may not be a best cost-effective solution. On the other hand, given that a video coding scheme is exploited with the consideration of specific features inherent in a target application, it will achieve a better trade-off between the computational complexity and coding efficiency because of its adaptive coding based on the image characteristics.

1.3 Existing Video Implementation Techniques

Table 1.2 Platform comparison.

Platform	Performance	Power Efficency	Flexibility	Choices of CODECs	Development Cost
Dedicated hardware	best	best	poor	-	higher
Reconfigurable hardware	better	better	good	limited choice	high
DSP or media processor	good	good	good	limited choice	medium
General purpose processor (e.g. PC)	medium good	poor	best	wide	low

With the formation of numerous coding algorithms, a number of implementation techniques have been proposed for a wide variety of video applications. These approaches are generally categorized into four groups shown in Table 1.2: general programmable processor, special-purpose programmable processor, reconfigurable processor, and dedicated hardware (Full custom LSI).

A programmable solution is that all of coding modules composed of video codec system are implemented by the software [13]. The software is then executed on a programmable implementation platform such as general-purpose processor (GPP) and special-purpose programmable processor like media processor.

The flexibility of implementation techniques is significant as video coding standards are continuously evolving along with the rapidly changeable user's requirements. Software implementation techniques provide the maximum flexibility among existing approaches since the software update or function modification can be readily performed using a easy-to-use programming tool. Again, programmable solutions enable to address the need of new applications quicker than full custom LSI. This derives from the fact that refinement of the algorithms and increase of their competitive edge can be accomplished by modifying the source code. Hence its development cycle period is shorter than that of hardware solutions so that it can meet the short time-to-market (TTM) requirement well.

Although GPPs have been used for a wide range of devices including PC, consumer electronics (CE) (set-top-boxes, video recorder etc.), personal digital assistance (PDA), cell phones, and so on, software solutions on GPPs suffer from the low speed and low power efficiency compared to hardware solutions. This is because their architecture of GPPs is not well suitable for video processing. Even though hardware acceleration units such as Intel's MMX/SSE2 [15] have been introduced to boost multimedia applications, limited register spaces of GPPs are still not suitable for very heavy dataflow task. Moreover, the latest video coding algorithms become more complex than their predecessors for ever-increasing coding efficiency, so that software solutions for them on GPP are hard to meet the deadline constraint of real-time applications. This is one of main reasons why complex video codec systems such as MPEG4 and H.264 are generally implemented by the hardware solutions.

To improve the performance of software implementation, special-purpose programmable processors such as media processors (MPs) or digital signal processors (DSPs) have been developed for multimedia applications. Media processor (MP) evolved from DSP is a platform targeted at digital media processing such as video and audio. Since MP utilizes the dedicated instruction sets and custom cache memories, it can achieve a faster speed than GPP. Moreover, for the sake of supporting various applications, MP provides configurability that allows selection of required functions from predefined elements such as cache size, optional instructions to improve functionality and performance through the addition of dedicated hardware modules. Furthermore, MP can better cope with rapidly changeable market requirements in the short time-to-market. This is because the instruction sets of the MP can be easily altered for new market requirements without any side-effect for other applications. On the other hand, if general-purpose processors do this, other applications will be affected due to their widespread application targets.

The popular media processors include media embedded processor (MeP) produced by Toshiba [3], Open Media Application Platform (OMAP) fabricated by Texas Instrument(TI) [16], and Blackfin provided by Analog Device (AD) [17]. It is well known that power efficiency is critical for product popularization of battery-powered devices such as mobile phone and laptop. In order to meet the power-saving requirements, additional software is needed for media processors to control the clock speed and voltage based on the tasks at hand. As a consequence, an additional overhead is added, albeit this approach can improve the power efficiency of media processors. Software implementation on Von Neumann-type processors tends to consume excessive power due to control-driven scheme and clock synchronized circuit, thus a more robust programmable implementation platform must be exploited for multimedia applications.

In order to meet the stringent requirements of real-time applications on mobile devices, hardware solutions have been adopted for video codec systems. This is due to the fact that they can achieve better performance and power efficiency than software solutions. In terms of the hardware solutions, both reconfigurable hardware and dedicated hardware are widely used for multimedia application fields so far.

The typical reconfigurable hardware implementation includes field-programmable gate array (FPGA) and complicated programmable logic device (CPLD). FPGA (or CPLD) consists of configurable logic blocks (CLBs) that implement various logical functions. For the sake of flexibility, the behavior of digital logic circuits can be described by a hardware description language (HDL) such as VHDL or Verilog. Next, the software description of a circuit can be synthesized using a synthesis tool, and then it can be programmed on FPGA chip that realizes the circuit. Moreover, modification of the circuit design can be realized by changing the HDL source code, and then modified hardware design can be reprogrammed on FPGA chip. In general, FPGA or CPLD design requires several weeks of engineering efforts instead of months because of no final fabrication process. Thus a product development cycle using FPGA is shorter than that of full custom LSI solutions. Nevertheless, video coding standards contain several complex computation modules such as motion estimation and compensation (ME/MC), so that a multimedia system built on reconfigurable hardware comprises a large number of basic logic units. As a consequence, it is hard to be embedded into the small ubiquitous devices such as cell phone and PDA.

On the other hand, a dedicated hardware implementation can overcome large-size challenge of FPGA. This is because they are hardwired and do not require programmable interconnection of basic logic circuits. Thanks to no programmable interconnection, a dedicated video codec system can achieve the highest speed and best power efficiency among the alternative solutions. Accordingly, full custom LSI can meet the demands of real-time applications on small ubiquitous devices well [13]. While they lack of flexibility and updatebility and extendability, because they are hardwired.

Due to above weakness points of full custom LSI, they hardly meet the users' changeable requirements. Moreover, with frequently updated situation of consumer electronics market, more and more used devices will become unserviceable items. Those devices left on our earth will be not good for our living environment. Furthermore, the full-custom LSI development of a design at the transistor level will require several years for design and testing. So long product development cycle can not satisfy the short time-to-market requirement. Therefore, a promising programmable platform with the high performance and power efficiency will be expected in the image/video coding application domain.

The availability of low-cost and low-power hardware with sufficiently high performance is essential for the popularization of image and video coding applications. To reduce the hardware cost of video codec system while keeping high performance and high power efficiency, a hybrid architecture such as SoC or SiP has been adopted. The SoC may ultimately offer the lower silicon cost than SiP. Thanks to on-chip I/O interface and on-chip memory, power dissipation and processing delay can be diminished too much so that it can meet real-time requirements on mobile devices. Furthermore, SoC system can be small enough to be embedded into portable devices due to integration of multiple processors. With the demands on much richer functionalities of consumer appliances, more and more processors will be executed on a single chip. This tends to bring several challenging issues including the design and verification of SoC system due to synchronized clocking circuits. Thus a promising architecture of SoC is demanded for the future multimedia applications.

1.4 Motivation of This Research

With the advance in multimedia and networking technology, digital video has been penetrated into a variety of application fields. For instance, the remote surveillance system has been emerging in some public places such as bank, airport and road as well as parking lots. Also, video chatting is very popular via internet or wireless network. It may be said that video coding has become an essential component of broadcast and entertainment media. To cater to this market demand, a number of video coding standards suitable for different application domains have been developed and commercialized. These modern coding techniques offer the possibility to store or transmit the vast amount of data necessary to represent digital images and video sequence in an efficient and robust way [1].

Since mobile devices suffer from weak computing power, short battery lifetime and limited display capability, standard video coding on such kind of small devices is usually realized by a dedicated hardware. Although dedicated video codec can achieve both fast coding performance and power efficiency, it is too inflexible and unextendable to satisfy the users' changeable requirements because they are hardwired. The software-based solution for image and video applications is therefore more attractive than the pure-hardware one.

In order to realize a fast coding by a software-based solution, a simple and efficient video coding scheme and its parallel software implementation platform are required for real-time applications on pervasive small devices. Such low complexity coding algorithm easily embedded with other various functions is demanded as its counterpart of generic compression standards such as MPEG4 [18] [19], JPEG2000 [20] and H.264 [6] [21]. This is because these standards endure too heavy computational load in spite of good reconstructed quality with a very low bit rate. Furthermore, those coding standards implemented on conventional programmable implementation platforms such as media processors or DSPs are hard to meet real-time requirements and power efficiency. This is because Von Neumann-type processors built on the control-driven scheme and clock synchronized circuit endure high additional overhead associated with context switching in case of the parallel processing. Since a data-driven processing platform based upon the data-driven scheme and self-timed super-pipelined circuit enables a large amount of parallelism without any additional scheduling code, it is more promising for the video coding software.

This thesis thus proposes a fast video coding scheme suitable for highly parallel software implementation, by simplifying both intra- and inter-frame coding in exchange for focusing on only semi-motion pictures observed in several video applications such as remote surveillance and person-to-person teleconference system. The proposed scheme is based upon block truncation coding (BTC). BTC is very simple but its image quality is not so good. In order to derive a better tradeoff between the image quality and computational complexity, an adaptive block truncation coding (ABTC) scheme using three-level classification and differential prediction coding modulation (DPCM) is introduced for both intra- and inter-frame coding. Furthermore, its highly parallel software implementation on a self-timed data-driven multimedia processor (DDMP) is discussed in terms of realizing programmable SoC systems. Finally, evaluation results illustrate that the proposed scheme can achieve the reconstructed image sequences up to 37 dB image quality and 60 compression ratio as well as 60 VGA f/s frame rate on average, which is twofold as that of previous scheme [22].

1.5 Organization of Thesis

Chapter 1 describes the research background and motivation. The first section introduces the video coding applications, main issues and metrics to evaluate a codec system. Then the existing video coding algorithms and their implementation approaches are reviewed, followed by the motivation of this research. Finally, the research framework is given in the last section.

Charter 2 discusses the requirements of a fast video coding on system-on-chip (SoC) for portable applications. In order to realize a fast coding on SoC, a simple and fast video coding scheme suitable for parallel implementation is necessary. Moreover, a parallel software implementation platform which can realize high-throughput performance and low-power consumption is required to realize a flexible and updatable video codec system on small portable devices.

Chapter 3 details the proposed ABTC scheme and its data-driven parallel implementation. ABTC is an extension of original block truncation coding (BTC), which is very simple but its image quality is not so good. In order to reach a better trade-off between the image quality and computational complexity, three techniques are utilized in the proposed video coding scheme. First, two optimal threshold values are used to realize the adaptive image coding based on the local image characteristics. Second, three coding approaches corresponding to the classified image blocks are adopted to reduce the computational complexity while keeping the good image quality. Last, DPCM is employed to improve the compression efficiency with little image distortion by virtue of inter- and intra-frame prediction. As for its data-driven parallel implementation, three techniques are utilized to achieve a desired throughput performance and delay time. The first is SIMD-type packet (two adjacent pixels are held in one packet). The second is associative temporal memory based on the tagged packet of dynamic data-driven scheme. The third is compound operators to increase the throughput of history sensitive processing.

Chapter 4 illustrates the experimental results of the proposed ABTC on data-driven chip multiprocessor (DDMP). First, the image quality of the proposed ABTC with the variable compression ratio is measured using three standard video sequences. The rate-distortion evaluation shows that the proposed ABTC is more suitable for the semi-motion pictures due to the 37 dB image quality with 0.4 bit rate, which can be observed in several applications such as person-to-person teleconference and remote surveillance. Moreover, the throughput performance and response time of DDMP implementation is evaluated. Two types of data structures are exploited and used in this experiment. It can be derived from the experimental results that the block-order input data is superior to scanline-order one due to twofold frame rate of the former one as many as that of the latter one. Moreover, response time of the block order is independent of the image size while that of scan-line order is increased along with the enlarged image. Thus it can be concluded that the block-order input data is more suitable for data-driven parallel implementation of ABTC scheme.

Chapter 5 summarizes the research work and presents the future perspectives. In this research, a fast and efficient video coding scheme and its data-driven parallel implementation have been presented for semi-motion picture applications. The experimental result has proved its coding effectiveness and efficiency. Moreover, the data-driven parallel implementation shows around VGA 60 f/s on average. These results illustrate that the proposed scheme implemented on data-driven processors can realize real-time coding with good video quality. In my future work, the coding efficiency of the proposed ABTC scheme will be further improved by a binary block partitioning method with variable-size block matching. Moreover, a circuit implementation for the proposed compound reference instruction of the line-buffer will be performed, then a practical video coding system on USB-DDMP will be established according to the proposed ABTC scheme. Furthermore, four future directions are outlined for this study which will be hoped to motivate the researchers to undertake this field.

Chapter 2

Requirements for A Fast Coding on SoC

2.1 Introduction

A simple and efficient coding scheme and its parallel software implementation platform are essential to realize a fast coding on flexible SoC. However, current video coding standards involve several computation-intensive functional modules such as ME/MC and DCT as well as entropy coding. This makes software implementation difficult to meet the deadline delay constraint required by real-time applications albeit these standards produce satisfactory image quality at a very low bit rate. A simple and efficient video coding scheme which acts as a counterpart of video coding standards is thus required to realize a fast coding on programmable SoC.

BTC [9] is a promising algorithm for realizing the software coding on the SoC by virtue of its simplicity and parallelism. While the basic BTC achieves a constant bit rate (2 bits/pel), it endures a limited application field. In order to improve the compression efficiency of basic BTC, there are several variants of BTC that have been proposed in the literature. These techniques include the vector quantization (VQ) [23], median filtering (MF) [24], discrete cosine transform (DCT) [25], and arithmetic coding (AC) [26].

In addition, modifications of BTC coupled with variable block size [27] or threshold technique [28] have been used to improve the image quality. Moreover, the latter is aimed at preserving the edges of objects. This is because BTC by its nature results in ragged edges and introduces noise at the edges. Furthermore, BTC can achieve a good reconstruction of the textured image, but it is not good at other image such as smooth image and smooth-variation image.

In order to realize a flexible SoC system, where several real-time multimedia applications can be concurrently executed along with users' requirements, it is important to develop a fast software implementation platform. There are two promising software implementation techniques so far. One is media processors targeted at multimedia applications on consumer appliances which have been produced by famous semiconductor companies. Some of representative media processors are media embedded processor (MeP) from Toshiba, open media application platform (OMAP) from Texas Instrument, Blackfin processor from Analog Devices. These media processors can realize the module configuration specific to target multimedia applications by virtue of custom instruction sets and memory modules. Owing to custom configurations, both the flexibility and desired performance can be accomplished by media processors.

Another one is microprocessor with multimedia instruction sets such as Pentium with MMX, ARM and SH. The latter ones (i.e. ARM and SH) are embedded microprocessor widely used in the mobile devices such as cell phone. By virtue of multimedia instruction sets, microprocessor is more powerful for multimedia processing. These approaches make real-time multimedia processing available in the today's embedded applications.

In the remaining sections, the current promising coding algorithms and existing promising software implementation platforms will be described to realize a fast coding on flexible SoC in more detail. The advantages and disadvantages of these algorithms and software implementation platforms are discussed. Finally, the issues of the existing promising schemes and implementation platforms are discussed in terms of a fast coding on SoC.

2.2 A Simple and Efficient Coding Scheme

A simple and efficient coding scheme is essential to realize a fast coding on flexible SoC. Block truncation coding (BTC) is a promising algorithm to obtain real-time video coding on the programmable SoC by virtue of its simplicity and parallelism. To improve the rate-distortion performance of the basic BTC, several modified BTC algorithms combined with other techniques have been proposed in the literature. These variations in some cases yield better performance than basic BTC.

To reduce the bit rate of basic BTC, several authors employed vector quantization(VQ) [23] [25] [29] [30] [31]. However, the size of code book has significant effects on the image quality and bit rate. In case of the large size of code book, good image quality will be accomplished with high bit rate. For the sake of reduction of bit rate, the small size of code book was selected in the emerging BTC-VQ schemes. For example, [23] and [25] use 256 code words. In [32], 64 masks are selected according to their edge-likeness. Moreover, two code books are utilized in [30], one is selected as described in [23] and the other is constructed to cover blocks with a single step edge intersection. When VQ is applied to both quantization data and bit plane, the bit rate is equal to around 1.5 bits/pel.

Furthermore, based on the fact that most of the redundancies in the bit plane exist in very close neighbors, [24] proposed a BTC hybrid coding coupled with the median filtering. Since the root signal space is smaller than the binary space, the number of bits of binary data is reduced by median filtering transform. In case that median filtering was employed for the binary block, the compressed data rate could reach around 1.375

bits/pel. Given that a binary block is coded with the trellis coding, a compressed data rate up to 1.1875 bits/pel will be achieved.

In addition to VQ and median filtering, a hybrid scheme based on BTC coupled with the discrete cosine transform (DCT) has been proposed in [25] [32] [33]. In [25], the DCT is used to compress the gray level information (i.e., the low and the high-low values). In [32], a smooth block is decided by comparing four low frequency DCT-coefficients in each of 8x8 blocks with the predefined threshold. In case of a smooth block, a DCT/VQ method is used. Otherwise, the block is divided into four, and then each sub-block is encoded by a BTC-VQ method [29]. Since DCT is a time consuming process, BTC combined with DCT will increase the decoding time albeit it enables the improvement of the rate-distortion performance. Consequently, this hybrid coding method dilutes the advantage of basic BTC, short decoding time.

To overcome the weak points of VQ and median filtering, [26] presented a modification of BTC in which the compression ratio was improved by arithmetic coding with an adaptive modelling scheme. The arithmetic coding is used to code the quantization data and the bit plane. The experimental results illustrate that entropy coding outperforms VQ [23] in both image quality and bit rate. Nevertheless, arithmetic coding endures so much computational complexity that practical implementation of this algorithm becomes difficult.

Although above hybrid BTC-based scheme coupled with complex coding techniques can improve the compression efficiency of basic BTC, their computational complexity is increased in that coupled techniques are time-consuming processes. Accordingly, except for the variations of BTC mentioned above, other modifications of the basic BTC seem as promising algorithms for establishing a fast codec system on flexible SoC due to their low complexity. These modifications of BTC will be described below in more detail.

In this section, how to simplify the coding schemes to enable a fast coding is discussed at first. Next, the existing promising block truncation coding schemes are described in detail, followed by issues in the current promising coding schemes.

2.2.1 How to Simplify the Coding Schemes

From the viewpoint of better trade-off between the computational complexity and compression efficiency, the approaches to simplify the coding schemes are summarized as follows.

The first, throwing away complex coding functions such as spatial-frequency transformation and entropy coding commonly used in the coding standards. Alternatively, a simple quantization method like mean and the first absolute moment is used in absolute moment block truncation coding (AMBTC) [35].

Sophisticated coding functions involved in the coding standards have robust capability to remove the redundancies of entropy information inherent in the image through each processing stage. By virtue of this capability, the coding standards can achieve better compression efficiency than the block truncation coding. Nevertheless, they are too heavy to realize real-time coding by software solutions on flexible SoC. Alternatively, block truncation coding employs simple quantization data which can preserve the local characteristics of image. It has been proved that such kind of coding scheme can obtain relatively good reconstruction because it takes advantage of the feature of human vision system. That is, human is more sensitive to the variance of textured image than that of less textured image.

The second, utilizing integer arithmetic computation without complex operators such as multiplication and square root to make coding process simple. As an example, AMBTC utilizes the first sample absolute moment AM rather than the standard deviation used in basic BTC. The AM can be computed by simple operators such as add and shift, whereas standard deviation contains the square root operator. Thus, AMBTC is much easier to be implemented than BTC due to its simplicity.

The third, employing an adaptive coding approach instead of the uniform one. Such kind of coding approaches enable encoding procedures adaptive to the characteristics of local images. Smooth regions are thus encoded using a simpler way in comparison of the complex area. In general, the threshold techniques are utilized to classify the whole image into different area. And then adaptive coding methods with diverse computational complexity are employed. Adaptive compression coding [28] is an example of this case.

The fourth, omitting the undetermined coding factors such as distribution of pixels for simplification of the coding process. For example, two-level reconstruction data of original AMBTC are computed using the number of pixels more than the mean value of block. In order to simplify the coding computation, an assumption is made that the number of pixels over the mean is identical to its counterparts. As a result, the number of pixels do not need to be considered when computing the reconstructed data at the decoder.

The fifth, taking advantage of the prediction approach to simplify the coding computation. This consideration is based upon inherent image features that the redundant information tends to exist in the neighboring pixels of the image and in the consecutive images. A prediction approach adaptive to the features of local image is used to examine the similarity of adjacent pixels, and then the current local image similar to its predecessor can be denoted by a predefined tag.

It should be noted here that the approaches summarized above are dedicated for block truncation coding for the sake of better trade-off between the compression efficiency and computational complexity. Accordingly, some of approaches such as prediction and classification techniques appended to the basic BTC will increase the conditional switching process, which tends to increment the coding procedures compared with basic BTC. While prediction and classification method can realize the image coding adaptive to local image features. This is to say, the smooth area is encoded by a simple way, while the complex area is encoded by a complicated way for reserving the more entropy information. As a consequence, such adaptive approach can obtain a better trade-off performance between the compression efficiency and computational complexity than a uniform coding way.

2.2.2 Existing Promising Coding Schemes

In this subsection, the existing promising coding schemes for realizing a fast coding on flexible SoC will be discussed. These promising schemes take advantages of some of the aforementioned approaches to simplify the coding procedures for availability of a fast coding while reserving a good compression efficiency.

(a) Basic block truncation coding

Block truncation coding is a lossy coding technique applicable for gray-scale images. That is, it reduces the file size but loses some original information of the image [9]. The significant advantages of this coding approach are low computational complexity and high parallelism.

The codec diagram of the BTC is shown in Fig. 2.1. It can be seen that BTC computes the mean and standard deviation (SD) over small blocks of input image. The equations for mean and SD are shown below.

$$\overline{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.1}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} |x_i^2 - \overline{x}^2| \tag{2.2}$$

where $\overline{x}$ refers to the mean of block image with the size of m = $N \times N$, x_i denotes the i_{th} pixel over the block image, and σ denotes the standard deviation (SD).

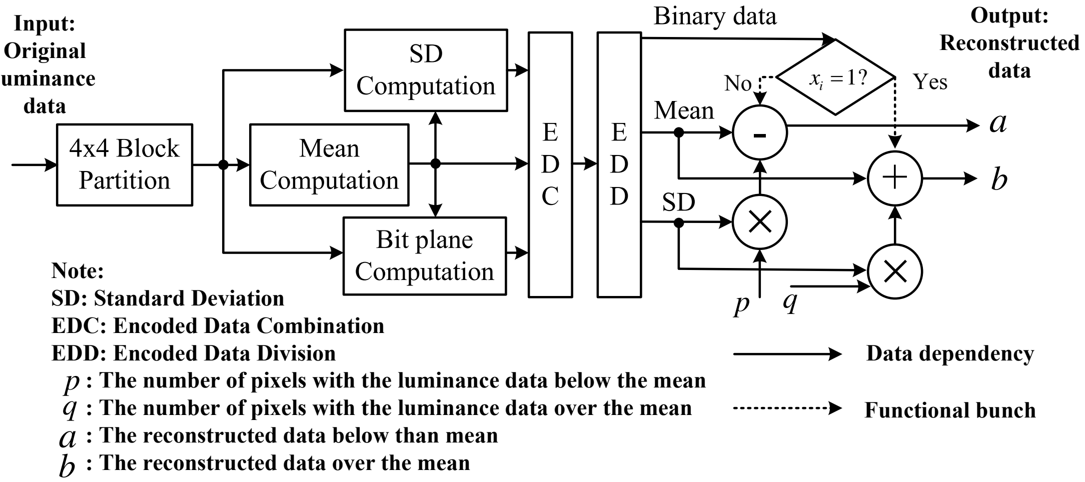

Input:
Original
luminance
data
4x4 Block
Partition
SD
Computation
Mean
Computation
Bit plane
Computation
E
D
C
E
D
D
Binary data
$x_i = 1?$
No
Yes
Mean
SD
-
+
×
×
p
q
Output:
Reconstructed
data
a
b
Note:
SD: Standard Deviation
EDC: Encoded Data Combination
EDD: Encoded Data Division
p : The number of pixels with the luminance data below the mean
q : The number of pixels with the luminance data over the mean
a : The reconstructed data below than mean
b : The reconstructed data over the mean
Data dependency
Functional bunch

Fig. 2.1 The diagram of BTC codec.

At the same time, by virtue of block mean as a threshold, each of small blocks is quantized into a one-bit plane to preserve certain statistical moment. The bit plane involves the binary data, where 1's represent the pixel data over the mean, otherwise, 0's. At the decoder, the two-level reconstructed data denoted as a and b are obtained by virtue of the mean, SD, and the number of pixels over the mean denoted as q. p is equal to $m - q$. a and b can be calculated below.

$$a = \overline{x} - \sigma\sqrt{\frac{q}{p}} \tag{2.3}$$

$$b = \overline{x} + \sigma\sqrt{\frac{p}{q}} \tag{2.4}$$

where a refers to the reconstructed data below the mean, otherwise, b. The bit rate of BTC is dependent upon the bit number of mean and standard deviation as well as the one-bit plane. As an example, for a 4×4 pel luminance block, if the mean and standard deviation are 8 bits, the bit-plane is 16 bits, then bit rate of BTC is 2 bits/pel.

In order to meet the requirements of bit rate less than 2 bits/pel, a modification of basic BTC is proposed using a 10 bits to jointly denote these two quantization data. This scheme is based on the observation that grey level quantization error is more visible in low variance regions [34]. Hence the mean of the pixel block is assigned more bits in blocks with small standard deviation and fewer bits in blocks with large standard deviation. In this way, the bit rate is reduced to 1.63 bits/pel.

(b) Absolute moment block truncation coding

Based on the basic BTC, an absolute moment block truncation coding scheme denoted as AMBTC [35] [36] [37] utilizes the first sample absolute moment AM in place of the standard deviation used in basic BTC. AM is obtained by the following equation.

$$AM = \frac{1}{m}\sum_{i=1}^{m}|x_i - \overline{x}| \tag{2.5}$$

At the decoder, the two-level reconstructed data are calculated as follows.

$$a = \overline{x} - \frac{m}{2p}AM \tag{2.6}$$

$$b = \overline{x} + \frac{m}{2q}AM \tag{2.7}$$

Using this new quantization method, AMBTC achieves better reconstructed image quality than the basic BTC. Furthermore, AMBTC has no square root and square computation, thus it is easier to be implemented than BTC. However, AMBTC still keeps high bit rate albeit reconstructed image quality of AMBTC is superior to basic BTC. Other variants of BTC transmit the lower mean $\overline{x}_{low}$ and higher mean $\overline{x}_{high}$ rather than the mean and absolute moment of block [25] [28] [30]. This technique can overcome the rounding error produced by the only computed $\overline{x}_{lower}$ and $\overline{x}_{higher}$ at the decoder. However, this method tends to increase the bit rate because the bit number of mean data is usually more than that of absolute moment AM.

$$\overline{x}_{lower} = \frac{1}{m-q} \sum_{x_i < \overline{x}} x_i \tag{2.8}$$

$$\overline{x}_{higher} = \frac{1}{q} \sum_{x_i >= \overline{x}} x_i \tag{2.9}$$

(c) BTC image coding with variable block size

As for basic block truncation coding, the block size is usually fixed 4 × 4, in order to avoid edge blurring and blocking artifacts in smooth areas. Nevertheless, using the fixed block size may not improve the compression ratio of smooth area such as background image. For this purpose, [27] proposed modified BTC with a variable block size called as vBTC.

In this algorithm, firstly a large size of the block image is input, and then the processing of basic BTC is performed block by block. Suppose that the standard deviation of the large size of block is over the predetermined threshold, then the size of the block will be divided by four until the standard deviation of block is less than the predefined threshold, wherein the minimal block size is 2 × 2.

Compared with fixed size of BTC, vBTC can offer better performance. The use of vBTC with optimal thresholds leads to a reduction of the error in the reconstructed image by almost 40% of the error in the reconstructed images obtained by fixed size of BTC [27].

(d) BTC image coding with thresholds

Targeted for the applications required edge preservation in the decoded image such as vision and robotics as well as recognition, [28] presented an adaptive compression coding based on the BTC and quad tree named as ACC. This algorithm is aimed for

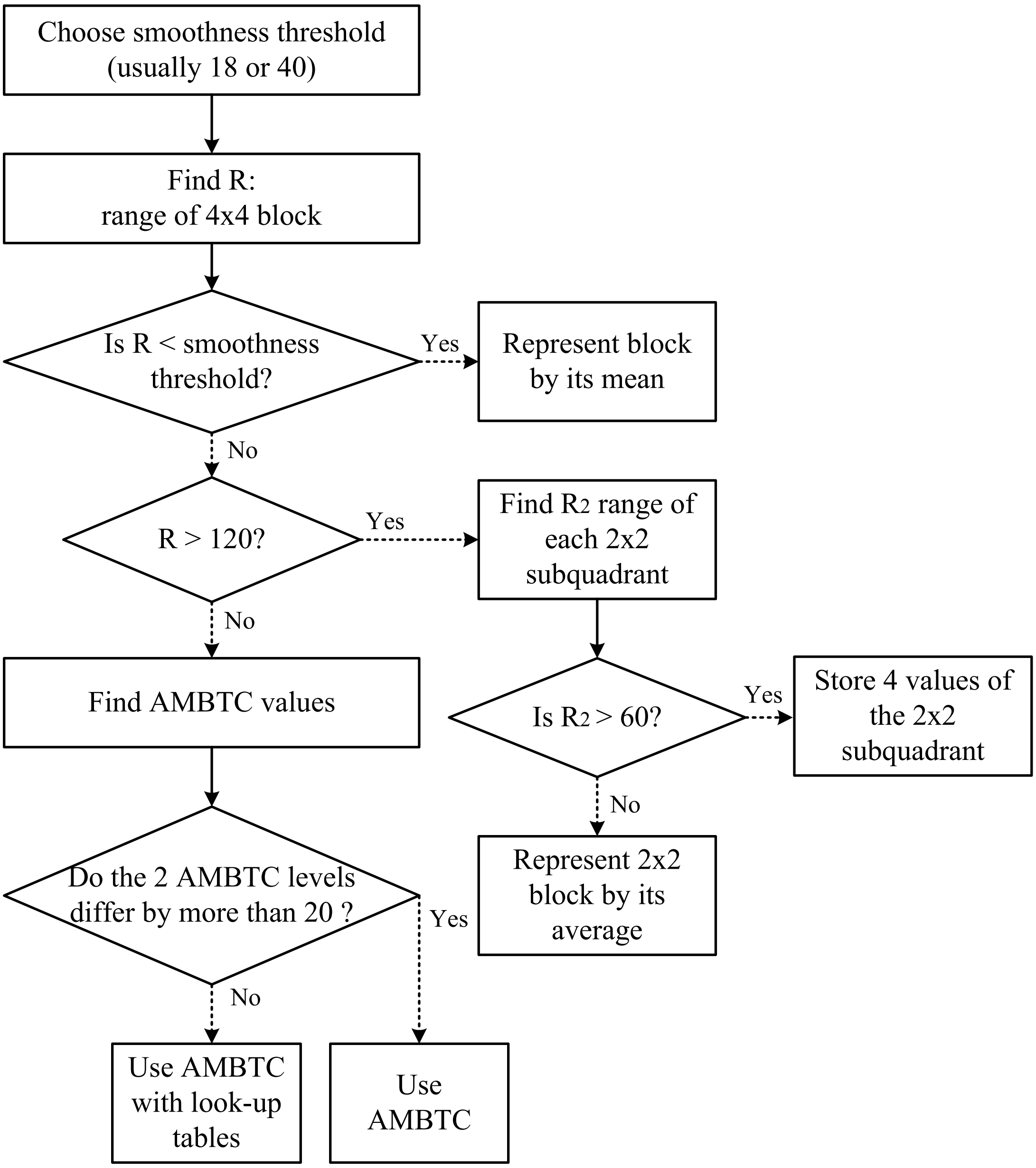

Choose smoothness threshold (usually 18 or 40)
Find R: range of 4x4 block
Is R < smoothness threshold?
Yes
Represent block by its mean
No
R > 120?
Yes
Find R2 range of each 2x2 subquadrant
No
Find AMBTC values
Is R2 > 60?
Yes
Store 4 values of the 2x2 subquadrant
No
Represent 2x2 block by its average
Do the 2 AMBTC levels differ by more than 20 ?
Yes
No
Use AMBTC with look-up tables
Use AMBTC

Fig. 2.2 The flowchart of ACC.

solving the problem that the basic BTC introduces the artifacts which cause a ragged, noisy appearance in regions where contain an edge. The flowchart of ACC algorithm [28] is shown in Fig. 2.2.

In order to realize an adaptive coding, two threshold values are utilized to identify a block image as one of three categories. Since this algorithm is aimed to preserve the edge, both thresholds are decided as the difference of maximum and minimum luminance value called as the range of block. The first threshold is named as the smoothness threshold, and the second one is referred to as the edge detection threshold. The best reconstructed image is produced by the smoothness threshold of 18 with the bit rate of 1.1-1.2 bits/pixel. On the other hand, better bit rate (0.7-0.8 bits/pixel) is obtained with the smoothness threshold of 40 as many blocks are represented by their mean.

Given that the range of the block is less than the smoothness threshold (18 or 40), then it is determined as the smooth block, which is represented by its mean. Otherwise, provided that the range of block is over the threshold, then the edge detection threshold is utilized to identify the block involved an edge. Suppose that the range of block is less than the edge detection threshold, the two-level reconstructed data are calculated by AMBTC. Then the difference between the two-level reconstructed data is computed, which is compared with the preset threshold 20. If this difference is less than 20, lookup tables coupled with AMBTC are used, otherwise, only AMBTC. If the range of the block is more than the edge detection threshold, the quad tree technique is used to identify the edge. Given the range of 2×2 block is more than 60, then 4 values of the 2×2 sub-quadrant will be stored in order to reserve the edge. Otherwise, 2×2 block is represented by its mean.

The experiment results have shown that ACC can preserve edges of pictures better than other variants of BTC without the quad tree method. Also, by virtue of the threshold technique, ACC realizes the adaptive coding to overcome the poor reconstruction in smooth region. Furthermore, this algorithm uses the lookup tables to code binary data block. By this way, the compression ratio is improved.

2.2.3 Issues in the Existing Promising Coding Schemes

Both the basic BTC and AMBTC are very simple but their image quality is not so good and bit rate is high, thus their application field is rather limited. This is due to the fact that the quantization threshold of each block image is only its mean value. Also, they do not consider the adaptive coding based on the image characteristics so that they can not realize good reconstruction of the less textured images. Although AMBTC achieves better image quality than the basic BTC, the improvement of image quality is very small.

Even though the variants of BTC coupled with other techniques such as variable block size and threshold (e.g. ACC) have been made significant progress in terms of coding efficiency, there are following issues having not been addressed so far.

The first, the most of current coding algorithms do not consider their implementation in total, so that their implementation is more complex even it is not reasonable for practical applications. Also, great engineering efforts need to be paid for the implementation of complex coding algorithms. This will result in increasing the development cost and delay time to the market.

The second, although BTC combined with variable block size can improve the bit rate according to the image features, but division of large block image into small block size based on standard deviation tends to increase the execution time. Moreover, this algorithm is not beneficial for parallel implementation because of its hierarchical data structure. Furthermore, this algorithm can not achieve better image quality due to just usage of the basic BTC.

The third, the adaptive compression coding (ACC) can realize the image coding adaptive to the image characteristics by virtue of two thresholds. Owing to the target applications of ACC, both of thresholds are determined as the difference of maximum and minimum luminance value over a 4 × 4 block. However, they can not represent the variance of luminance of a block image, in spite of better edge reservation than other algorithms. Also, ACC does not consider the high correlation between the neighbor pixels, the efficient prediction methods are thus not used. Nevertheless, the prediction methods can efficiently remove the inter-block redundancies so that the significant compression gains of basic BTC can be obtained with the negligible image distortion. In addition, ACC is very complex in that many techniques such as quad tree and lookup table are utilized to realize the adaptive coding for the sake of better edge reservation. ACC can not thus realize an optimal tradeoff between the image quality and computation complexity.

Since the issues mentioned above are involved in the existing promising coding algorithms, a more robust coding algorithm which can realize a better trade-off performance between image quality and computational complexity must be developed.

2.3 A Fast and Parallel Software Implementation Technique

In order to meet the requirements for continuously emerging applications, the various software implementation platforms have been developed and applied to current multimedia appliances. The flexible SoC is an effective and robust implementation technique for the decrease of the fabrication cost of the current consumer electronics products.

Even though today's CPUs continue to keep pace with Moore's law, traditional programmable processors do not have architectures that are well suitable for video processing [14]. This is due to the fact that video processing includes block-based and pixel-level tasks. Such dataflow-intensive tasks require the large register space which are not included in the traditional CPU and DSP architectures. In order to meet the real-time video applications, there are two approaches to speed up the video processing. One alternative approach is to introduce the hardware acceleration units through intrinsic instructions; examples of such instructions include Intel's MMX/SSE. Another alternative approach is the media processors. In this section, the software implementation techniques for realizing a fast coding on SoC are discussed in corporation with the examples of the multimedia processing architectures.

2.3.1 How to Realize Fast and Parallel Software Processing

In order to enable fast and parallel software processing in the multimedia applications, a variety of implementation tactics have been developed so far. These tactics are based upon the unique features of media processing, including large parallelism, repetition of operations and intensive memory access. According to these features, the current implementation approaches could be classified into following three types.

The first is the exploration of parallelism with different levels. These parallelism comprise data-level parallelism (DLP), instruction-level parallelism (ILP) and control-level parallelism (CLP). Data-level parallelism exploited has been employed in the complicated instruction sets computers (CISC). Examples of processors in this category are the Intel's Pentium with MMX/SSE units [15] and D30V from Mistubishi [38]. Instruction-level parallelism can be exploited with very long instruction word (VLIW)/SuperScalar(SS), or super pipelining. VLIW-based examples are MAJC from Sun Microsystems and FR500 from Fujitsu [13]. Processors with SS and pipelined feature include the new IA-64 based processor [39] family from Intel and HP and AMD's Athlon. Control-level parallelism (thread-level parallelism) leads to speculative thread execution. MAJC [40] and FR500 are good examples for this approach [13].

The second is the specialized hardware support for repetitive operations such as multiply-accumulation computation (MAC), block search, summation of absolute difference (SAD) etc. The examples for this category are MAP100A [41] and TANGRAM [42]. This approach is very similar to providing macros for a group of successive instructions in general processors.

The third is the usage of on-chip memory or high-speed link between the CPU and the off-chip memory. To use on-chip memory is aimed to reduce the times of off-chip memory access as the off-chip access is slow and large-power process. The example for on-chip memory is IRAM [43] to reduce the need for transfers by embedding a significant portion of the memory onto the chip. The example for the latter is the Pentium 4 based 850 chipset with a high speed RDRAM (Rambus Dynamic Random Access Memory).

In addition, the out-of-order execution logic is used in the intel's Pentium processor in order to diminish the side-effect causing by the delay of some instructions. The processor with out-of-order execution logic attempts to find as many ready independent instructions to execute as possible, even though they are not in the original program order. While using this logic, the additional retirement logic unit is required to recover the original program order of instructions reordered by the out-of-order logic.

The mentioned-above approaches used in the general- or special-purpose programable processors have empowered the fast video codec for today's real-time multimedia applications. In the following subsection, the current state-of-the-art software implementation platforms are delineated in corporation with the examples.

2.3.2 Existing Promising Software Implementation Platforms

The current general-purpose programable processors adopting some of aforementioned approaches are generally categorized into two groups, RISC processors and CISC processors. Moreover, to meet the the low-power and real-time communication requirement of the mobile electronics, special-purpose processors dedicated for media processing has been developed according to the inherent features of media processing. The examples of the media processors are Multimedia Embedded Processor (MeP) from Toshiba, Open Media Application Platform (OMAP) processor from Texas Instrument and Blackfin processor from Analog Devices.

In this subsection, the general-purpose programmable processors including RISC and CISC processors will be introduced, followed by the application-specific programmable platforms with the benchmark processors.

(a) General-Purpose Programmable Processor

General-purpose programmable processors developed for widespread applications today find a niche in the multimedia applications by adding the special instruction sets. Based on the architecture of special instruction sets, the general-purpose programmable processors can be grouped into RISC and CISC processors.

(I) RISC processor

The instruction set of the RISC processor is characterized by the most frequently used instructions for general-purpose computing. More complex instructions or less frequently used instructions are implemented as a sequence of the reduced instruction set [13]. The new generation of processors such as the Sun UltraSPARC (termed as super scalar) [44] [45] [46] can process up to four instructions simultaneously per cycle so as to realize the instruction-level parallelism. To cater to the emerging multimedia applications, the SPARC family of microprocessors implement the SPARC ISA (Industrial Standard Architecture) version 9, a 64-bit ISA with a multimedia extension called VIS [47]. These instructions are used for the specialized pixel operations that can operate in parallel on 8-, 16-, or 32-bit integer values packed in a 64-bit floating point (FP) register [13]. Moreover, the concept of sub-word parallelism has been exploited in several other RISC-based designs to enhance the media processing [48] [49] [50]. Other leading vendors and designers of RISC processors with support for the media processing include ARM processors, PowerPC's AltiVec and the PA-RISC 2.0 architectures [51].

The latest ARM [52] offers two complementary technologies for media acceleration: ARM OptimoDE data engine technology, and the recently announced NEON technology. The optimoDE data engine is a configurable VLIW-style DSP architecture targeted at intensive non-stop data processing, such as video processing. OptimoDE data engines provide a very high level of performance with a very efficient implementation. In general, each design is tuned to an application domain, and can be reprogrammed to handle different implementation of the applications. For example, an OptimoDE data engine designed for video can be reprogrammed by software designed to handle MPEG4, H.264 without changing the base hardware design.

ARM NEON technology is a SIMD architecture extension to the ARM processor architecture that is targeted at flexible media and signal processing. NEON technology provides powerful media acceleration within a general-purpose architecture. Each design can have its implementation matched to the desired performance level of the ARM core. Microarchitecture details such as memory system interfaces and execution units are left to the specific implementation, while programmer's model remains the same across all designs. NEON technology accelerates a wide variety of applications-including video, audio, and 3D graphics-without the need to modify the core. This provides the capability for simultaneous acceleration of different media types, and gives the flexibility to handle new applications after the design has been deployed.

ARM with these two new technologies is promising for future all-in-one-chip applications in the smart phones. This is due to the fact that configurable OptimoDE data engines enable ARM to provide the highest domain-specific processing performance within a small gate count and minimal power budget. Moreover, ARM coupled with NEON technology will enhance a high level of general-purpose signal processing capability. Accordingly, ARM processors combined with these two technologies are promising to meet the future demanding multimedia processing requirements of the embedded market.

Another representative RISC processor for embedded applications is SH from Renesas Technology Corp [53]. SH-4 [54] achieves 360 MIPS and 1.4 GFPLOPS at 200 MHz. Its architectural enhancement is based on two-way superscalar issues of instructions, sperate instruction and data caches, and early stage branches. The unique floating-point vector instructions are effective for 3D graphics processing. SIMD extension involved in the latest SH-5 [53] provides the parallelism required for efficient execution of a wide range of applications including home video games and handheld PCs. Also, SH-Mobile application processor is used for mobile phone system. The new software, the SH-mobile V2 (type name SH7310) and SH-Mobile3 (type name: 73180), incorporates an MPEG4 full hardware accelerator for the third-generation (3G) mobile-phone audiovisual communications.

(II) CISC processor

Compared with RISC processors, the CISC processors incorporate with more complicated and feature-rich ISAs to enhance the media processing capability. An example for the CISC processors is Intel'S Pentium with MMX/SSE. MMX/SSE extension to the Intel architecture is designed to accelerate the multimedia and communication software running on the Intel architecture processors [55]. MMX instructions are 64-bit packed integer SIMD operations that operate on 8-, 16-, or 32-bit operands. The SSE instructions are 128-bit packed IEEE single-precision floating-point operations. The

Pentium 4 processor adds new forms of 128-bit SIMD instructions called SSE2. The SSE2 instructions support 128-bit packed IEEE double-precision SIMD floating point operations and 128-bit packed integer SIMD operations. The packed integer operations support 8-, 16-, 32-, and 64-bit operands [15].

The intel's Pentium-4 processor [15] has a FP execution cluster which executes the FP, MMX, SSE, and SSE2 instructions and utilizes the following features.

(1) Many FP/multimedia applications have a fairly balanced set of multiplies and adds. The FP adder can execute one extended-precision (EP) additions, one double-precision (DP) addtion, or two single-precision (SP) additions every clock cycle. This gives a peak six GFLOPS for SP or three GFLOPS for DP FP at 1.5 GHz.

(2) Many multimedia applications interleave adds, multiplies and pack/unpacked/shuffle operations. For integer SIMD operations, which are the 64-bit wide MMX or 128-bit wide SSE2 instructions, there are three execution units that run in parallel. The SIMD integer ALU execution hardware can process 64 SIMD integer bits per clock cycle.

(3) A separate shuffle/unpack execution unit can also process 64 SIMD integer bits per clock cycle. MMX/SSE2 SIMD integer multiply instructions use the FP multiply hardware mentioned above to do a 128-bit packed integer multiply op every two clock cycles.

(4) The FP divider executes all divide, square root, and remainder micro-operation. It is based on a double-pumped SRT radix-2 algorithm, producing two bits of quotient (or square root) every clock cycle.

The intel's processor coupled with MMX/SSE can obtain 1.5 to 2 times performance gains compared with the one without MMX/SSE instructions. Since the multimedia processing requires the access of data from off-chip memory, the long latency will occur due to frequent access of off-chip memory. In order to reduce long latency of memory, the deep buffering of Pentium 4 processor (126 *μops* and 48 loads in flight) allows the machine to execute large sections of the program to examine the dependencies. Moreover, the out-of-order execution hardware often unrolls the inner execution loop of these programs numerous times in its execution window. This dynamic unrolling allows the Pentium 4 processor to overlap the long-latency FP/SSE and memory instructions [13].

Along with the extension of multimedia applications of general-purpose processors, the application-specific processors (i.e. media processor) dedicated for media processing have been proposed for mobile multimedia processing. The state-of-the-art media processors will be discussed in the following parts.

(b) Application-Specific Programmable Processor

Media processors evolved from the digital signal processor (DSP) which is generally for audio signal processing are developed for today's and tomorrow's multimedia applications on portable consumer appliances. Since the architecture of media processors is designed based upon the features of media processing, they tend to achieve a better cost/performance ratio than the general-purpose processor. The state-of-the-art media processors contain the multimedia embedded processor (MeP) of Toshiba [3], Open Media Application Platform (OMAP) processor of Texas Instrument (TI) [56] and Blackfin processor of Analog Devices (AD) [57].

(I) Multimedia embedded processor (MeP)

MeP [3] from Toshiba is a platform for digital media SoC chip that is targeted at the digital media processing applications including video and audio. The MeP core composed of configurable 32-bit CPU is a kind of microprocessor for embedded applications and can be used in the same way as the ARM9 and MIPS32 4K. The biggest difference is that the MeP is user configurable. The features and instructions can be selected to be implemented in each MeP core from a set of options at design time. As an example, the multiply-and-add and absolute difference instructions are added to reduce processing time or change the instruction RAM size according to the program code size.

MeP can be developed in a hierarchy way for different applications. A MeP core coupled with one or more extension units forms a MeP module. The extension units contain the user-custom instruction (UCI) unit, DSP unit, coprocessor and hardware engine. For instance, one of the modules in MPEG2 codec, is tailored for video codec applications. Specifically, a user custom unit and hardware unit are added to support MPEG2 processing, including variable length coding and decoding, discrete cosine transform (DCT)/inverse DCT, quantization and inverse quantization, as well as motion compensation.

Asymmetric multiple MeP modules are composed of a SoC chip. Since they are each tailored to particular multimedia processing like video, audio or graphics, MeP SoC chip has smaller area and less power dissipation in comparison with general-purpose processor where runs any software on a single large processor. Moreover, MeP enables the implementation of complicated functions in a short time by virtue of flexible reuse of the intellectual property (IP) blocks. Hence, MeP can accomplish the short time-to-market performance required by the consumer electronics market.

(II) Open media application platform (OMAP) processor

OMAP [16] from Texas Instrument (TI) is a highly integrated hardware and software platform designed to meet the application processing needs of next-generation embedded devices.

The OMAP5912 [56] application processor is composed of TMS320C55x DSP core and ARM926EJ-S RISC processor. The C55xDSP architecture achieves high performance and low power through increased parallelism and total focus on reduction in power dissipation. The C55x CPU provides two multiply-accumulate (MAC) units, each capable of 17-bit x 17-bit multiplication in a single cycle. A central 40-bit arithmetic/logic unit (ALU) is supported by an additional 16-bit ALU. Use of the ALUs is under instruction set control, providing the ability to optimize activity and power consumption. The ARM926EJ-S is a 32-bit processor core that performs 32-bit or 16-bit instructions and processes 32-bit, 16-bit, or 8-bit data. The core uses the pipelining so that all parts of the processor and memory system can operate continuously. By virtue of the dual-core architecture, OMAP can desperate the multimedia processing tasks onto each of core based on the feature of tasks. Accordingly, OMAP can achieve better cost-effective performance than the general-purpose processors due to its adaptive processing.

(III) Blackfin (BF) processor

Blackfin processor [57] from Analog Devices is a 16/32-bit embedded processor designed to meet the computational demands and power constrains of today's embedded audio, video, automotive, industrial/instrumentation, and communication applications. Blackfin processors combine RISC MCU and DSP functionality.

MCUs are traditionally used for asynchronous control flow, and DSPs are well for synchronous, constant-rate data flow such as audio or voice-band applications. This integration of MCUs and DSPs can thus enable better performance than either of them. This is because each optimization is performed according to the feature of different tasks. DSP applications usually focus on performing as many arithmetic computations as possible in the fewest number of core block cycles. To this end, DSPs often use complex VLIW instructions, which erode code density and can increase code memory requirements. To enhance the multimedia processing capability, DSP often couples the ancillary processors to make up for a lack of flexibility. By virtue of integrating RISC microcontroller (MCUs) and DSPs into a single processing platform, BF processors can eliminate the need for separate digital signal and control processors, which reduces bill of material costs and greatly simplifies hardware and software design tasks. BF processor can therefore achieve a high-performance and low power dissipation and short time-to-market performance.

(IV) Features of media processor

Based on the aforementioned discussion, the key features of the architecture of the state-of-the-art media processors are summarized as the following points.

The first, the availability of high performance supporting the video processing via customized configuration such as memory size and instruction sets. As an example, the MeP core consists of basic core and configuration parts including memory size, instruction sets, debug support unit, and interrupt controller.

The second, parallel multi-processor architecture, where each of modules is designed for a specific application. For instance, a module contains processor core and extension units such as a VLIW (very long instruction word) coprocessor. With this configuration, the three instruction can be executed in parallel.

The third, efficient and flexible programmability. This is due to the fact that the extension units can be customized for specific application such as video codec or audio codec based on the user's requirements. As an example, the MeP provides the optional configurations which enable the designer to build a system suitable for target applications. Moreover, MeP SoC is composed of many MeP modules, each of modules is dedicated for a special application such as a video codec or audio codec. The module composed of the SoC can be used for future other SoC.

The fourth, low power dissipation suitable for embedded applications. Since the each module composed of SoC is configured according to the target application, it is capable of the low power dissipation and high performance. As an example, a flexible SoC is an asymmetrical multi-processor platform, so that it can achieve similar processing performance to the general processor with less clock speed. As an example, the MPEG2 codec built on the MeP requires 150 MHz with power consumption of 2 watts, while over 3.6 GHz with power consumption of around 20 watt is required in case of the general-purpose processors.

2.3.3 Issues in the Existing Promising Implementation Platforms

General-purpose processors with specialized media extended instruction sets can speedup the media processing with the exploration of different-level parallelism. While these processors employ a dynamic branch prediction technique, which implies the need for large primary and secondary level caches. This occupies valuable silicon area and proves disadvantageous during cache misses due to real-time constraints that need to be met for the media applications [13]. Moreover, the architecture of general-purpose processors is not optimal for multimedia processing due to its single processing core, albeit it provides the maximum flexibility. Generally, the single core can not realize processing adaptive to the feature of the different tasks. As a result, video stream processed by general-purpose processors requires the higher clock speed than special-purpose media processing platform. Moreover, higher clocking speed will result in the more power dissipation. Consequently, general-purpose processor is not suitable for battery-supported mobile devices.

The application-specific programmable platforms such as media processors can overcome the weak points of general-purpose processors by virtue of the customized module architecture. Each module composed of a media processing platform is responsible for a preset task. For example, the specific instruction sets frequently used in the media processing application such as MAC and SAD, are added by hardware solutions. While such customization of each module endures limited flexibility compared with the general-purpose processers.

Moreover, the programmable implementation platforms mentioned above are based on the conventional Von Neummn control-driven mechanism and synchronizing clocking circuits, so that the processing power enhanced by the increased clock rate will result in the clock skew problem. In addition, the synchronized clocking circuits endure the long wiring length across the chip, which imposes the extra power consumption. Since the portable applications supported by battery require the better power-saving processing performance, such as mobile phone and laptop PC, a new architecture of flexible software implementation platform is required. Data-Driven multimedia processor is promising to overcome above problems owing to its data-driven scheme and self-timed super-pipeline circuits. Data-driven scheme enables no context switching whether parallel execution of a program for a different set of instances or concurrent execution of different programs [58].

A self-timed data-driven architecture is promising for future video codec systems because of its data-driven scheme and self-timed local hand-shaking. A data-driven scheme empowers parallel execution of a program for a different set of instances or concurrent execution of different programs with no overhead associated with context switching. This is because in the data-driven processors, the operation is to be fired only depending on a pair of data necessary for execution of the operation [58]. This feature enables high throughput performance required for real-time multimedia applications. Furthermore, a self-timed clocking scheme empowers autonomous power-saving features by local hand-shaking. Thus, a data-driven processor coupled with self-timed super-pipelined implementation is promising for future battery-supported multimedia applications.

2.4 Discussion

Since the more and more applications require the audio, video and communication processing capability, low-cost and high-performance multimedia systems are demanded to boost the prevalence of the rich-featured consumer appliances. SoC is an optimal solution for reducing the system cost and better cost-effective performance due to its high integration of multiple processing units.

In this chapter, the requirement for a fast video coding on flexible SoC was discussed in conjunction with the promising existing algorithms and programmable processors. To realize a fast coding on SoC, a simple coding scheme and its fast and parallel software implementation platform are required.

Such kind of coding scheme with low complexity and high parallelism enables a fast codec system which flexibly cooperates with other real-time functions required by the users. BTC and its existing variants are promising schemes for realizing a fast coding on SoC, while they can not accomplish the better trade-off between the image quality and computation complexity well. Therefore, a more promising video coding scheme is required for this target.

As for its software implementation, the existing media processors are promising for real-time multimedia applications by virtue of their custom configurations. As an example, Toshiba's MeP disperses the intensive processing over multiple processing cores, each of which possesses the customized configuration based on the features of executed tasks. Hence, the less clock speed is required to accomplish the multimedia processing tasks in comparison with the general-purpose processors. While the customization of each module composed of SoC will result in the limited flexibility, in that they are each optimized for very different tasks. In addition, many functions have been integrated on the single chip along with the demand of rich-featured consumer electronics. This will lead to the productivity challenge with regard to the conventional Von-Neumann processing platform based on clock-driven scheme and synchronized circuit implementation.

To resolve the aforementioned issues, a more promising platform is required for future embedded applications. Data-driven chip multiprocessor is a promising platform for realizing a fast coding on flexible SoC by virtue of its data-driven scheme and self-timed super-pipelined circuits.

Concerning conventional Von Neumann-type processors, parallel implementation of a program or concurrent execution of different programs will incur rather high overhead associated with the context-switching [58]. On the other hand, data-driven processors can overcome this disadvantage. This is due to the fact that the execution of operations is only dependent on the available of a pair of data required for this operation. This computing paradigm enables more power-saving performance in comparison with the control-driven paradigm due to no centric control.

Furthermore, the self-timed super-pipelined implementation enables better throughput performance required by media processing applications. Synchronous pipeline circuits composed of current media processors require the additional intricate controls for pipeline flushing and interlocking as well as bubble suppression alongside the pipeline [58]. These controls affect the throughput performance and result in additional power consumption. On the other hand, self-timed super-pipelined implementation enables superior throughput performance to the synchronous pipeline implementation because of no intricate control.

In particular, the self-timed super-pipelined implementation makes the circuit verification easier than synchronized clocking one. This is because electromagnetic effect within an integrated SoC becomes more significant than before along with the incoming of deep-micron fabrication technology. Although highly integrated SoC empowers smaller size and more functions, this tends to result in a big challenge for circuit verification. While a self-timed super-pipelined circuit composed of pipeline stages has no systematic bus. This leads to that circuit verification can be performed based on each pipeline stage. Accordingly, this approach is capable of the decrease of the engineering efforts so as to shorten the time to market.

In order to realize a fast video coding on flexible SoC, a fast video coding scheme suitable for highly-parallel software implementation is proposed in the following chapter. Moreover, its highly-parallel software implementation on the data-driven processing system is discussed for flexible SoC.

Chapter 3

Adaptive BTC on Data-Driven Processing System

3.1 Introduction

This chapter therefore proposes a fast video coding scheme suitable for semi-motion pictures and then describes its data-driven highly-parallel software implementation. The proposed coding algorithm is based upon the absolute moment block truncation coding (AMBTC) [35]. AMBTC calculates the mean of each block and then performs a two-level quantization so that it is very simple but the image quality is not so good. In order to improve the image quality, a AMBTC-based coding scheme has been described in [22], while in this thesis the proposed scheme has significant features which are different from the previous scheme. First of all, in order to derive a better trade-off between reconstructed quality and computational complexity, the proposed scheme introduces a three-level classification technique. Compared to the previous two-level classification technique presented in [22], the proposed three-level classification technique enables more adaptability in encoding/decoding an image. Moreover, to further improve the compression efficiency, differential pulse coding modulation (DPCM) is employed in the current scheme. DPCM is utilized to remove the redundant information existing in neighboring block images within an identical image. By virtue of these coding techniques, i.e. three-level classification and DPCM, the proposed scheme can achieve better rate-distortion performance than the previous one [22]. Furthermore, its data-driven parallel implementation is discussed to realize a fast video codec on a data-driven chip multiprocessor implemented by the self-timed super-pipelined circuit [58].

The structure of this chapter is as follows. In the next section, the proposed adaptive BTC scheme is described in detail. In Section 3.3, its data-driven parallel implementation is discussed to realize a fast coding on a data-driven chip multiprocessor. Finally, the features of the proposed ABTC on the data-driven processing systems and the principles of the load balance on a data-driven multiprocessor are discussed in Section 3.4.

3.2 Adaptive BTC Scheme

Most of the existing block truncation coding schemes presented in the literature such as [12] [24] don't care about their implementation in total, while in this research both a coding scheme and its implementation are taken into account as an integrated system. When an adaptive BTC scheme [60] was designed, simplification and parallelization of the original BTC [9] were performed simultaneously in order to realize a fast coding on the programmable SoC as well as to guarantee the reasonable image quality.

The proposed adaptive BTC scheme (ABTC) [61] is illustrated in Fig. 3.1. As shown in Fig. 3.1 (a), an input RGB image is first partitioned into 4x4 unoverlapped block images, followed by the color space conversion from RGB to YCrCb for better coding efficiency. Next, the Y components go through AMBTC encoder. After that, the inter- and intra-frame DPCM are used to remove the temporal and spacial redundancy inherent in consecutive frames and neighboring pixels in the identical image,

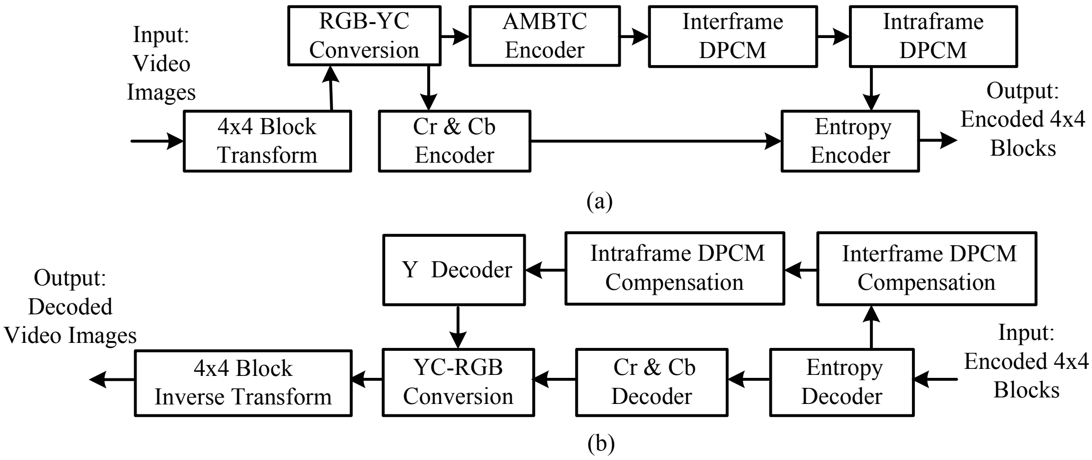


Fig. 3.1 Block diagram of the proposed ABTC scheme: (a) Encoder; (b) Decoder.

respectively. At the same time, the C components (Cr and Cb) are encoded in that they are independent from Y component. Since human is more sensitive to luminance changes than chrominance variance in a picture, mean value of every chrominance block is merely calculated and transmitted. Finally, the encoded data and side information are combined using entropy encoder before storage or transmission.

The ABTC decoder is shown in Fig. 3.1 (b). It can be seen that the procedures of the decoding is reverse to that of the encoding shown in Fig. 3.1 (a). While the decoding process is simpler than encoding one because of no computations of some parameters such as AM and SAE. This research therefore focuses on the ABTC encoder.

Regarding the decoding process shown in Fig. 3.1 (b), whether this block belongs to the one similar to its counterpart in the previous frame (SPF) is first checked by a one-bit flag. Given this bit is 1, then this block is SPF. The corresponding decoded block in the previous frame is used to compensate the current SPF block. Otherwise, the following two-bit flag is checked to identify the type of encoded block image. Suppose this two-bit flag is 11, then this block is the one similar to its previous block in the same image (SPB). SPB is compensated by its previous decoded block in the identical image.

In the remaining subsections, the intra-frame coding of the proposed ABTC scheme is first described. Moreover, a square root quantization method used to encode the pattern blocks is presented. Finally, a simple BTC-based interframe prediction is introduced.

3.2.1 Intra-Frame Coding

(I) ABTC algorithm

The intra-frame coding of the proposed adaptive BTC scheme shown in Fig. 3.2 is mainly composed of several pipelined modules: AMBTC encoder, DPCM for three types of block images. A stream of 4x4 luminance block images are input into AMBTC encoder, and then mean value ($\overline{x}$) and absolute moment (AM) are calculated by AMBTC as follows:

$$\overline{x} = \frac{1}{l}\sum_{i=1}^{l} x_i \tag{3.1}$$

$$AM = \frac{1}{l}\sum_{i=1}^{l} |x_i - \overline{x}| \tag{3.2}$$

where x_i denotes the i_{th} pixel value in a block of $l(= 4 \times 4)$ pixels. A one-bit plane which contains 1's where $x_i < \overline{x}$ and 0's where $x_i \geq \overline{x}$ is used to retain local properties of the image.

Considering the characteristics of the useful images, in which only a small region contains high detail or texture such as people and building, thus each luminance block is classified as either a uniform block, a normal block or a pattern block based on the predefined thresholds of AM and mean absolute error (MAE) calculated as follows.

$$MAE = \frac{1}{l}\sum_{i=1}^{l} |x'_i - x_i| \tag{3.3}$$

where x'_i is the i_{th} pixel value in a reproduced block image. Nevertheless, MAE is

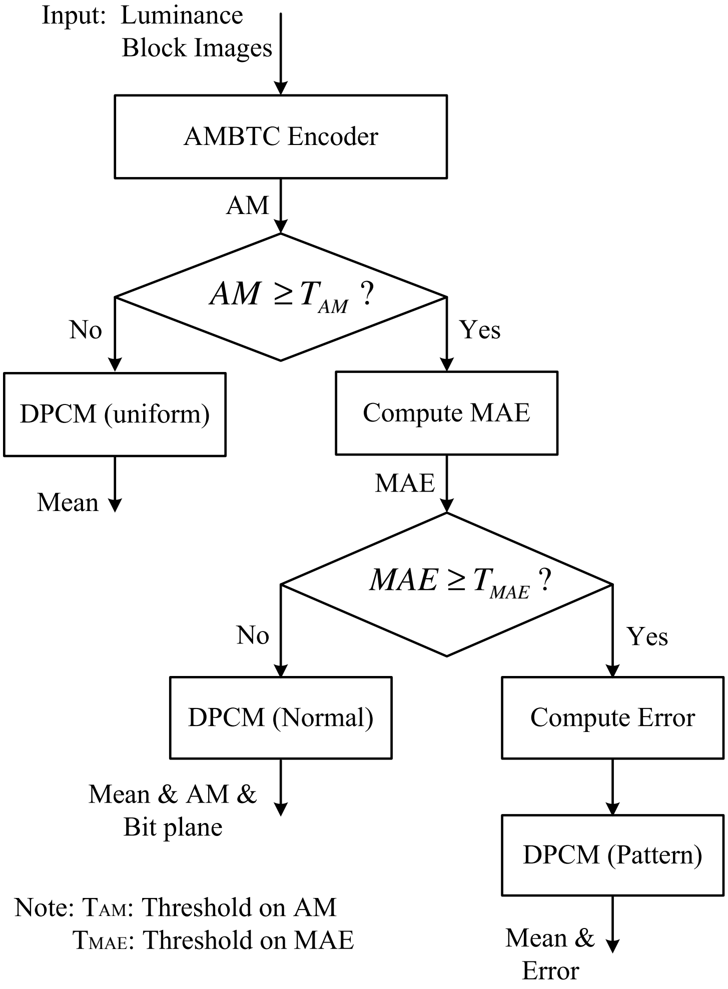


Fig. 3.2 The flow chart of the proposed algorithm.

usually replaced by SAE (summation of absolute error) in the practical implementation for decrease of operations. If the AM of a luminance block is less than the threshold value, then this block is referred to as a uniform block; if the AM of a luminance block is over the threshold and also MAE is less than the threshold, then this block is named after a normal block, otherwise, a pattern block. Moreover, each sort of block is labelled by a 2-bit identifier.

In order to derive a better trade-off between the reconstructed quality and computational complexity, those types of block images mentioned above are encoded by distinct approaches. As for a uniform block, the mean of a block image is merely used to reproduce the image at the decoder because encoding a smooth block using its mean results in a good reconstructed image. As for a normal block, three moments, mean, AM and bit-plane are calculated by simplified AMBTC without consideration of the distribution of pixel values over each normal block image. Then the encoded data are transmitted. At the decoder, the two-level decoded data, x_l and x_h, are computed as follows.

$$x_l = \overline{x} - AM \tag{3.4}$$

$$x_h = \overline{x} + AM \tag{3.5}$$

As for a pattern block, the mean errors are first computed by the equation (3.6). In order to obtain high image quality while reserving the low bit rate, an adaptive approach is used to determine the bit number of the mean errors. That is to say, the bit number of the mean errors is dependent upon the maximum value among 16 mean errors of each 4x4 pattern block. Moreover, the mean errors are first truncated based on the desired bit rate performance. And then the truncated mean errors and mean value over a pattern block are transmitted.

$$E_i = x_i - \overline{x} \tag{3.6}$$

where E_i is a mean error of the pixel value in a pattern block. The sign of the mean error is reserved by the one-bit plane. At the decoder, the reconstructed data are obtained as follows.

$$x'_i = \overline{x} + E'_i \tag{3.7}$$

where E'_i is a truncated mean error of the i_{th} pixel value in a pattern block.

To further improve the rate-distortion performance of the ABTC scheme, DPCM is introduced to remove the redundancies of entropy information inherent in the neighboring blocks. DPCM is derived from the fact that adjacent pixels possess a high degree of correlation within a picture, which is attributed to the increase of the compression ratio through comparing adjacent block images. Suppose that the current block is similar to its previous one, then only a two-bit identifier is transmitted.

In order to reduce the computational complexity with negligible image distortion, three arbitral approaches corresponding to aforementioned block images are adopted to identify the similarity between the adjacent block images. As for two adjacent uniform blocks, the difference of means (*difMean*) of block images is merely computed by the following equation and used for an arbitrator.

$$difMean = Mean_i - Mean_{i-1} \tag{3.8}$$

If *difMean* is over the predefined threshold, then the mean of current block is transmitted. As for two contiguous normal blocks, three moments, *difMean*, difference of AM (*difAM*) and difference of bit-plane map (*difMap*), are calculated by the equations (3.8), (3.9) and (3.10), respectively. These moments are used to examine the similarity between the block images within a picture. Moreover, only if all moments are smaller than the predefined threshold values, the current block is judged as the same to the previous block.

$$difAM = AM_i - AM_{i-1} \tag{3.9}$$

$$difMap = \sum_{j=1}^{l} (Map_{i,j} \bigoplus Map_{i-1,j}) \tag{3.10}$$

where i refers to the i_{th} block, j denotes the j_{th} pixel in a pixel block, Map represents a one-bit plane, $difMap$ denotes the number of different bits between the adjacent bit planes.

On the other hand, as for two adjacent pattern blocks, two parameters, $difMean$ and summation of absolute difference (SAD) calculated by equation (3.11), are utilized to decide the similarity between the pattern blocks.

$$SAD = \sum_{j=1}^{l} |E_{i,j} - E_{i-1,j}| \tag{3.11}$$

where $E_{i,j}$ denotes the mean error of the j_{th} pixel over the i_{th} pattern block; otherwise, $E_{i-1,j}$ denotes the mean error of the j_{th} pixel over the $(i-1)_{th}$ pattern block. Moreover, only if both $difMean$ and SAD is less than the predefined thresholds, then the current block is identified as the same to its previous one.

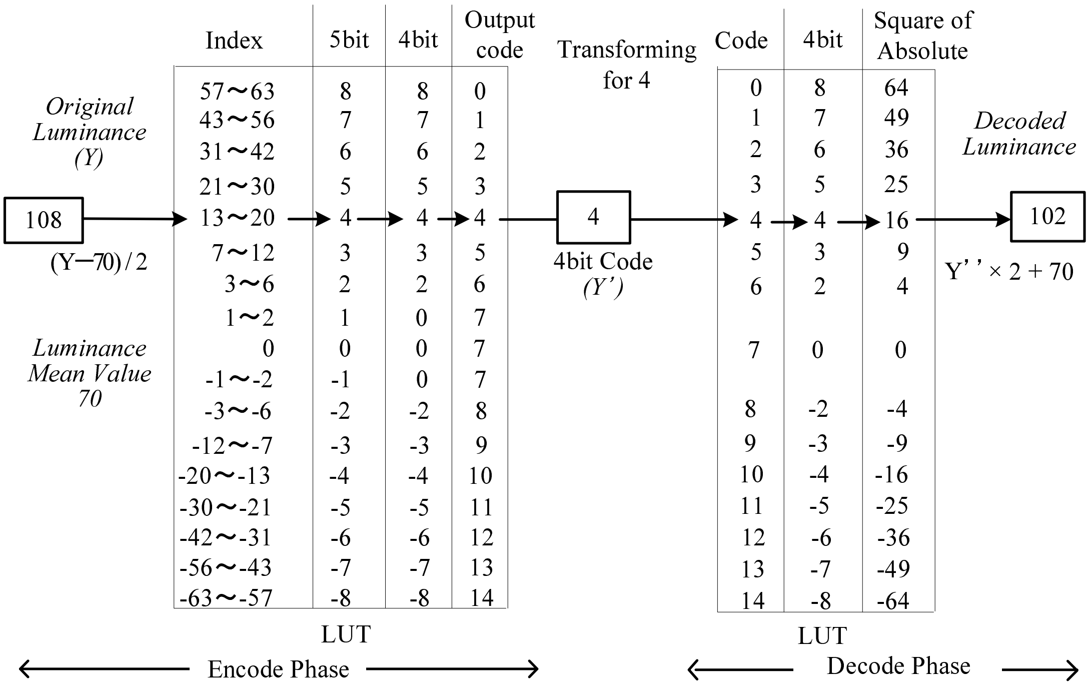
(II) Square Root Quantization for Pattern Block

In order to decrease the bit rate of ABTC algorithm, [63] proposed square root quantization (SRQ) approach to encode mean errors of each pattern block. The proposed square root quantization (SRQ) is formulated as equation (3.12). The input luminance value Y is first subtracted the mean value of Y ($\overline{x}$) and then the absolute difference of Y and $\overline{x}$ is computed, wherein one bit is used as the sign bit of (Y-$\overline{x}$). Next, the absolute difference is divided by 2 and the result is quantized into a value in the index table. After that, square root value of the index value is obtained. The obtained value requires 5-bit, thus it is transformed into 4-bit value for less bit rate. At the decoder, the process is inverse of the encoding procedure.

$$Y_i' = Sign(Y_i - \overline{x}) \times Round\sqrt{\frac{|Y_i - \overline{x}|}{2}} \tag{3.12}$$

Where Y_i' is the encoded Y, round denotes that fractional number of square root is round to the nearest integer number. Sign indicates the symbol of each mean error.

Using above scalable quantization, each 8-bit mean error in the pattern block is transformed into 4-bit code including one-bit sign. In order to eliminate the complex quantization computation, a lookup-table implementation is employed for the aforementioned quantization.



Index	5bit	4bit	Output code
57∼63	8	8	0
43∼56	7	7	1
31∼42	6	6	2
21∼30	5	5	3
13∼20	4	4	4
7∼12	3	3	5
3∼6	2	2	6
1∼2	1	0	7
0	0	0	7
-1∼-2	-1	0	7
-3∼-6	-2	-2	8
-12∼-7	-3	-3	9
-20∼-13	-4	-4	10
-30∼-21	-5	-5	11
-42∼-31	-6	-6	12
-56∼-43	-7	-7	13
-63∼-57	-8	-8	14

Code	4bit	Square of Absolute
0	8	64
1	7	49
2	6	36
3	5	25
4	4	16
5	3	9
6	2	4
7	0	0
8	-2	-4
9	-3	-9
10	-4	-16
11	-5	-25
12	-6	-36
13	-7	-49
14	-8	-64

Fig. 3.3 Lookup table implementation.

Fig. 3.3 shows an example of SRQ encoding and decoding process. At the encoder, mean value of a 4 × 4 pattern block is first calculated. In this example, mean value is 70. One of luminance values in the pattern block is 108, then 108 is subtracted 70 and divided by 2. The result is equal to 19, then 19 is approximated into 16 in the index table to make the deviation as small as possible. After that, the square root of 16 is equal to 4. Based on this idea, a lookup table (LUT) is designed including 256 codewords each of which is 4-bit. Taking advantage of this lookup table, a 4-bit codeword corresponding to one mean error can be directly obtained by a LUT operation. In this case, mean-error luminance value is positive, then the sign is 0. At the decoder, input encoded data is

4, square 4 is 16. Next, 16 is multiplied 2, and the product is equal to 32. Moreover, 32 is added 70, the summation is 102. Thus, when the encoded data 4 and mean value equal to 70 are input, 102 is outputted via a LUT operation.

3.2.2 BTC-Based Interframe Prediction

Motion prediction is very important to obtain a satisfactory image quality and desired compression ratio. The well-known block-based motion estimation and compensation technique is widely utilized in the current video coding standards such as H.263/H.264 and MPEG2/MPEG4. While this motion prediction technique is time-consuming process, it is too heavy to realize software implementation for real-time applications. Although a number of fast motion estimation approaches have been reported [64] [65] [66] [67], which can achieve better results than that of conventional fast search algorithms, such as three-step search, the 2-D logarithmic search, the conjugate direction search [68]. While these fast motion estimation schemes are not well compatible with our intra-frame coding. Therefore, a simple and efficient prediction approach is required for a fast coding on flexible SoC. To this end, a simple and adaptive motion prediction approach which can be harmonious with the intra-frame coding of ABTC scheme is presented in this subsection. This is because the entropy moments produced by AMBTC are taken full advantage of to justify whether the current block is a motion block.

In the conventional motion-estimation approach, large block size is adopted (i.e. 16x16 or 8x8 block). In this case, if the entropy moments of one block image are utilized to justify its status, large distortion in the reconstructed image will be occurred. In addition, the shape of motion objects is not block so that large block size results in unsatisfactory image quality, albeit the low bit rate performance may be obtained. In order to overcome this weakness, the latest video coding standard such as H.264 uses the variable block size, while the computational complexity of motion prediction is increased drastically, although the optimal compression efficiency can be accomplished.

In the proposed motion estimation approach referred to as DPCM, small block size of 4x4 is employed to remove the temporal redundancies existing in the consecutive frames. Owing to small block size, utilizing the moments of a block image as the arbitrary factors can produce a more precise motion prediction. Moreover, the motion prediction method is adaptive to the type of blocks. This is aimed to reach a better trade-off between the computational complexity and image quality. In fact, it is a simple block-based difference coding as current block is only compared to its counterpart in the previous frame.

The proposed approach shown in Fig. 3.4 includes different processes according to the types of two block images involved in the consecutive frames. If both are uniform blocks, the mean values are only compared. Suppose that $difMean$ is over a certain threshold, current block is decided as a motion block. This is because comparing two smooth blocks using their means leads to more accurate motion prediction. On the other hand, if one of them with AM over the predefined threshold exists, then three parameters, $difMean, difAM, difMap$ are compared, respectively. This is because examining a high-detail block using its mean can not accurately ascertain its status. Moreover, only if all parameters are smaller than the predefined threshold values, the current block is decided as a stationary block. In this case, no still block is encoded and transmitted at all. These threshold values can be obtained adaptively depending on the user's requirements of either BPP (bit per pixel) or signal-to-noise ratio (SNR) or both.

The encoded data structure for classified blocks is presented in Fig. 3.5. It can be observed that only 1 bit is demanded to represent the block similar to its counterpart in the previous frame (SPF), and three bits are required to denote the block similar to

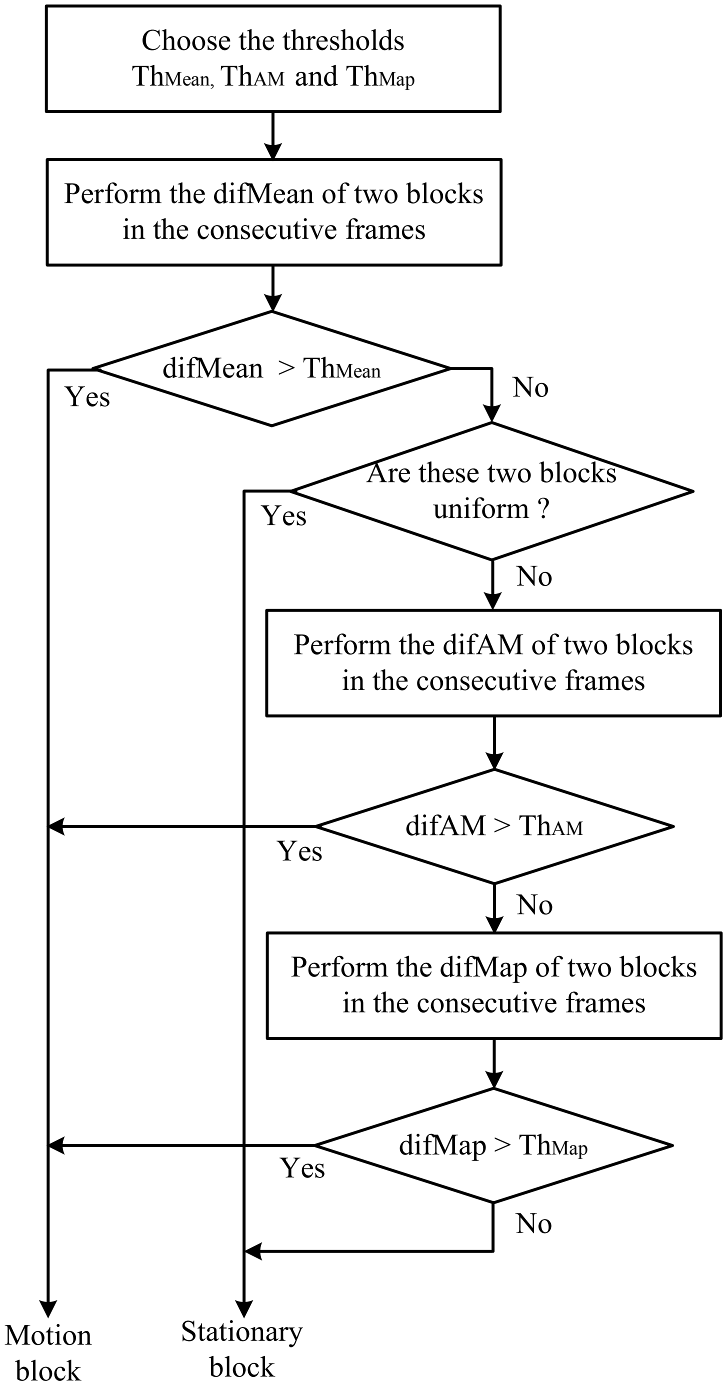
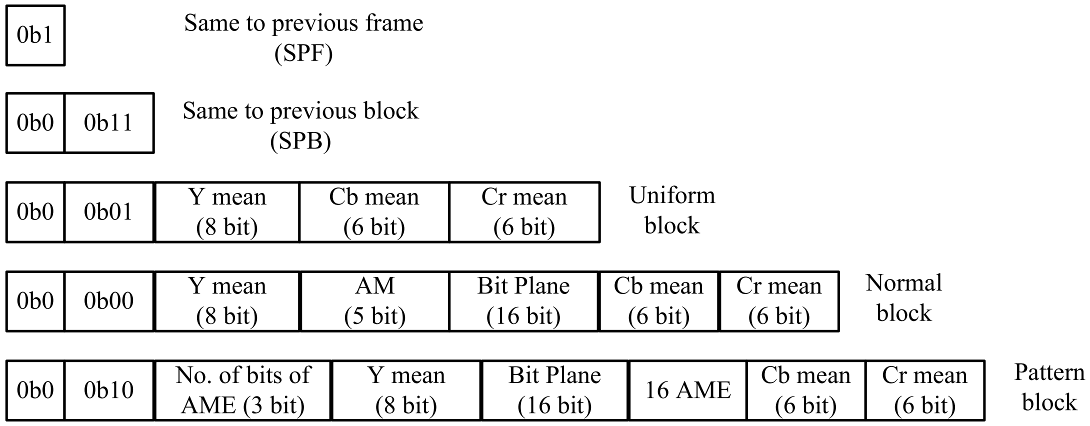

Choose the thresholds
ThMean, ThAM and ThMap
Perform the difMean of two blocks in the consecutive frames
difMean > ThMean
Yes
No
Are these two blocks uniform ?
Yes
No
Perform the difAM of two blocks in the consecutive frames
difAM > ThAM
Yes
No
Perform the difMap of two blocks in the consecutive frames
difMap > ThMap
Yes
No
Motion block
Stationary block

Fig. 3.4 Flowchart of a block-based difference coding.



Note: AME (Absolute value of Mean Error)

Fig. 3.5 Data structure of various block images.

its previous block in the same image (SPB). The compression ratio (CR) for each type of block images can be computed as follows.

$$CR = \frac{16a}{N} \tag{3.13}$$

where a refers to as the number of bits for each original pixel, while N represents the total number of bits for the encoded data of each 4x4 block image.

Since the input video sequence is colored into RGB images in the experiments, a is equal to 24 bit. Accordingly, the CR of SPF is equivalent to 384, while that of SPB is equal to 128. Concerning a uniform block with N=23 bits, CR=16. Otherwise, for a normal block with N=44 bits, CR is equal to around 8. While the CR of a pattern block is dynamic, as the number of bits of its encoded data is dependent upon the maximum value of mean errors (ME).

It can be deduced from Fig. 3.5 that the compression ratio of a pattern block is the least among these predefined blocks due to 16 mean errors (ME). While the number of pattern blocks is generally small because a common image includes small high-detail regions such as people and building. Especially, the DPCM is involved in the proposed scheme, which can further decrease the number of pattern blocks. Hence, the desired compression ratio can be obtained using the proposed ABTC.

3.3 Data-Driven Parallel Implementation

The dynamic data-driven chip-multiprocessor [58] is a promising platform for the proposed ABTC software because it allows us to process high-bandwidth stream data under the lower electric power by virtue of unique features of the data-driven scheme and the self-timed pipeline hardware.

In the data-driven processing scheme, each operation indicated by a node is not to be fired until a set of data necessary for the operation exist on the all input arcs of the node. Moreover, in the dynamic data-driven scheme, parallel execution of a program for different sets of data instances can be allowed without context switching and predesignated scheduling. Thus multiple processors based on the dynamic data-driven scheme can efficiently cooperate and their throughput is not hindered by latencies caused by memory access, processing delays, and interprocessor communication delays. Nevertheless, in conventional Von Neumann-type processors where instructions are executed in a predetermined sequence, such kinds of latency significantly affect the rate of execution of sequential instructions. In the data-driven processors, on the other hand, flow rate of the pipeline is completely independent of both total delay in the pipeline and the length of the pipeline. The data-driven processors therefore can achieve better pipeline throughput performance than the conventional processors.

In order to utilize its pipeline processing capability, it is essential to maximize the data flow rate in the pipelines of all the processors. This leads to the scalable implementation along with the available number of processors. To keep maximum data flow in each processor, it is necessary to not only reduce the number of operations but also exploit the parallelism inherent in it.

Moreover, the data-driven processing systems enable the concurrently parallel process. By virtue of the concurrent process, the response time of the history sensitive process can be reduced too much. Accordingly, the desired throughput performance can be accomplished. Concurrently parallel implementation associated with the history sensitive process will be discussed in subsection 3.3.3.

3.3.1 Parallelism Inherent in ABTC

Parallelism inherent in ABTC are exploited in a hierarchical way. Given the data dependency exists between the elements, then the pipelined parallelism can be exploited. Otherwise, concurrent parallelism. Based on the luminance data dependency, three-level parallelism are defined here. The first is a coarse-grained level which refers to the frames of a video sequence. The second is a medium-grained level which refers to the blocks of a frame. The third is a minimum-grained level which refers to the pixels of a block.

On the coarse-grained level, there are data dependencies existing in the consecutive frames of a video sequence because of the inter-frame prediction of the ABTC scheme. Accordingly, the pipelined parallelism can be exploited for this level. Likewise, on the medium-grained level, pipelined parallelism can also be exploited owing to the intra-frame prediction. As shown in Fig. 3.1, different block data can be simultaneously processed in the temporal dimension so that the pipelined parallelism are fully realized. On the fine-grained level, pixel data within each block image are independent of each other, the concurrent parallelism can therefore be fully exploited. Again, luminance component has no relationship with chrominance one, concurrent parallelism can be exploited.

It should be noted here that the exploited parallelism may exceed the allowed maximum of the available hardware resources. In this case, the processing efficiency of a data-driven processor will be affected or it can not work at all. Accordingly, the exploited parallelism must be customized into the available hardware resources. Customized methods for one single processor include the reduction of the number of concurrent processing nodes and the decrease of the input flow rate. The former is derived from that the limited capacity of the matching memory composed of a data-driven processor. The latter arises from the limited number of pipelined stages within a data-driven processor.

3.3.2 Two Types of Input Data Structure

Two types of input data structures shown in Fig. 3.6 are explored. Fig. 3.6 (a) illustrates that a stream of data with original raster scanline order are input into the ABTC encoder. For such input data structure, the throughput of block data fluctuates between maximum ($= \frac{wh}{4t_0}$, where w and h are the width and height of a frame image; t_0 is interval time between pixel data) and minimum ($= \frac{wh}{(3w+4)t_0}$). Fig. 3.6 (b) indicates that ABTC encoder receives a sequence of 4x4 block images, whose throughput is equal to $\frac{wh}{16t_0}$. So that the block-order input enables a constant flow of data stream. This is due to the fact that there is no three-line delay time arising from the original scanline order but an additional transform process is required.

To transform the original scanline-order image into a sequence of 4x4 block images, a compound reference instruction of the line buffer is proposed and shown in Fig. 3.7. Before utilizing this proposed instruction, the original pixel data with scanline order is stored into the line buffer based on the coordinates of pixels. Moreover, the line buffer requires the least capacity of 5120 (640x8) words for VGA image to ensure the steady data flow. As shown in Fig. 3.7, input data is the number of the last line over each 4x4 block image. The three offsets of buffer address are 3, 2 and 1 in case of 4x4 block images. Subtracting each offset from the number of last line, the data address of its

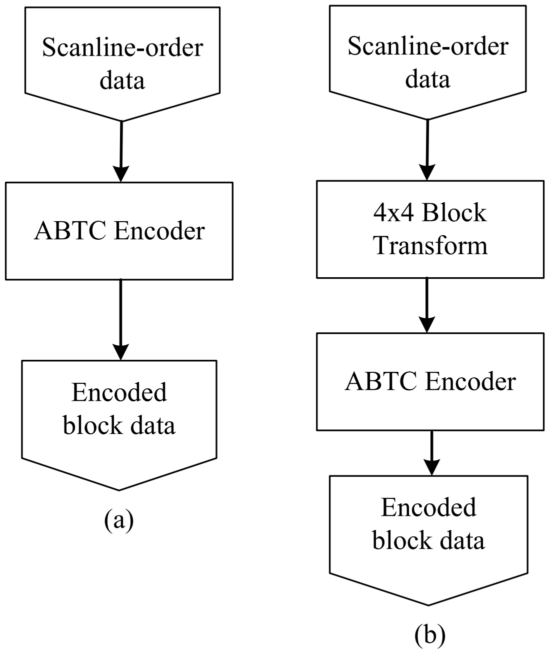


Fig. 3.6 Input data structure: (a) Scanline-order input; (b) Block-order input.

prior line can be obtained. Then three data belonging to its prior lines over a 4x4 block are read and output. The block transformation of every frame can be thus speedily realized and target throughput performance will be achieved by virtue of this proposed compound instruction.

3.3.3 Pipelined Parallel Implementation

Since functional modules composed of ABTC codec have distinctive computational characteristics, it is important to analyze their processing natures and then to perform a superior pipeline implementation for better systematic throughput performance.

Based on the essential data dependencies, each functional module in the ABTC codec can be identified as either a history sensitive process or a functional process. Since the throughput performance of a history sensitive process is an inverse ratio of the response time of the critical path, it is necessary to minimize the length of the critical path in order to maximize the throughput performance.

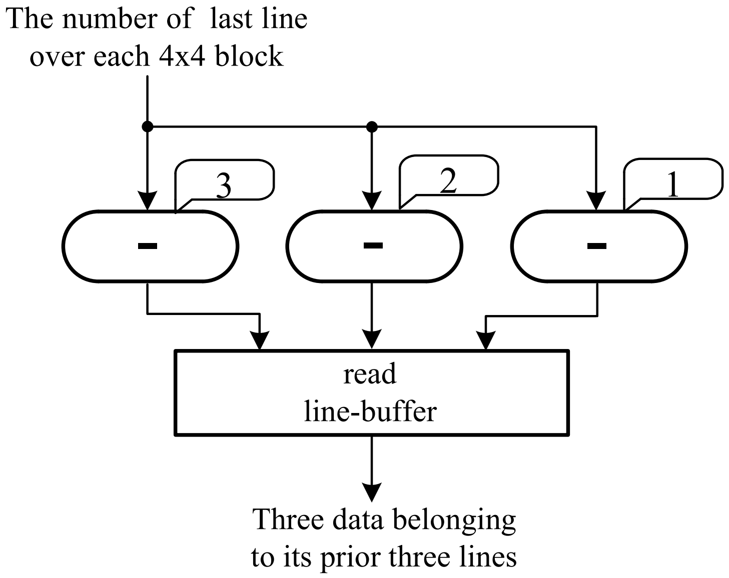


Fig. 3.7 A compound reference instruction of the line buffer.

As an example, Fig. 3.8 (a) illustrates a dataflow graph that sums up 16 input pixels of a block and calculates its mean by a 4 bit right-shift operation. In this case, an intermediate accumulated sum is fed back to the add operator repetitively. This kind of feedback path is generally involved in the history-sensitive process like Fig. 3.8 (a), and it might be a critical path, i.e., the longest critical path affects the total pixel rate of the ABTC program. The execution time of a partial program belonging to the longest critical path in the ABTC program therefore must be reduced.

Fig. 3.8 (b) shows that our data-driven implementation by which the feedback path is distributively stuffed into each compound operator (*read* & *add*) so that the execution time of the critical path can be minimized at the software level. In this sophisticated implementation, the following techniques are introduced: First, SIMD-type data packet (two pixels can be held in a packet). Second, associative temporal memory based on tagged packet of dynamic data-driven scheme. Third, compound operators: *read* & *add*, *swap* & *add* & *shift*.

The data-driven program shown in Fig. 3.8 (b) accepts a stream of 8 packets each

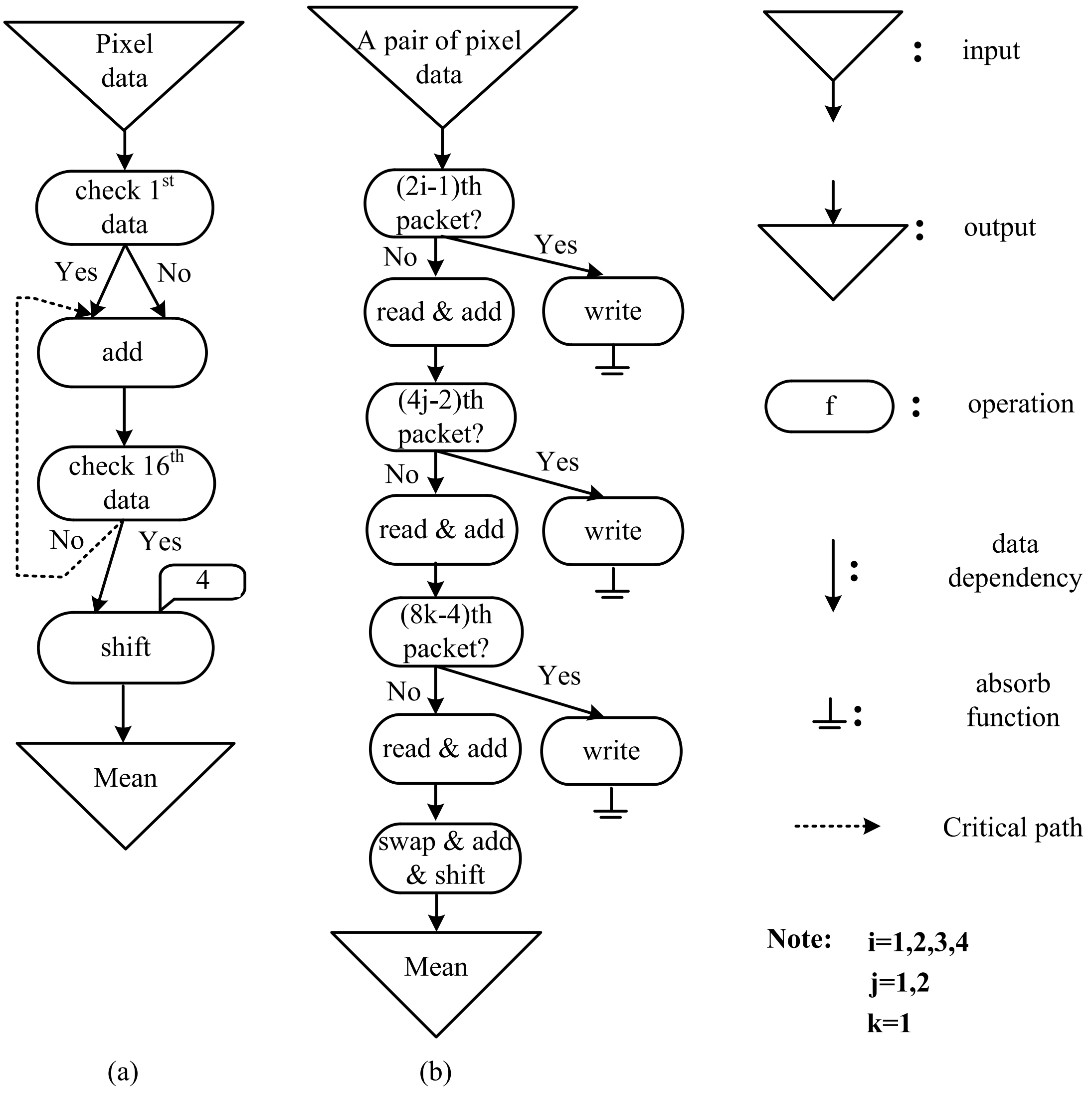


Fig. 3.8 Mean module: (a) Data dependency; (b) Data-driven implementation.

of which holds two neighbor pixels in a 4x4 block. This program is unfolded to a normal concurrent program which sums up 8 packets by a tree structure of binary adders. At the beginning of the program, the $(2i-1)$th $(i = 1, 2, 3, 4)$ packet is stored into the temporal memory with its identifier. Then the $(2i)$th packet goes to the 1st *read & add* operator and two intermediate sums are calculated from the $(2i-1)$th and $(2i)$th packets. At the 2nd *read & add* operator, the $(4j-2)$th $(j = 1, 2)$ packet is read and added to the $(4j)$th packet in the same way. After that, the $(8k-4)$th $(k = 1)$ packet is read and added to the $(8k)$th packet. Finally, two intermediate sums in the 8th packet are added and shifted, and then the final mean is outputted.

To make a quantitative comparison of throughput performance, response time (T_R) in Fig. 3.8 (a) and Fig. 3.8 (b) was computed, respectively. T_R is a product of critical path length and response time of each instruction (t_c). It can be derived from Fig. 3.8 (a) that T_R is equal to $15(t_c + t_c) + t$, where t denotes the time of the second pixel in a block image arriving at *add* operator. In our pipeline implementation shown in Fig. 3.8 (b), suppose that $t^{'}$ is the time of the 8th packet arriving at *read* & *add* function, then response time $T^{'}_R = t^{'} + 6t_c$. In case of DDMP, $t^{'} = 7t$, $t_c = 42t$, thus $T_R/T^{'}_R$=4.9. This ratio shows that throughput of Fig. 3.8 (b) is around 5 times as many as that of Fig. 3.8 (a).

As for functional process, its throughput performance is dependent on both number of operations and available processors. Hence the number of operations must be decreased so as to maximize the throughput performance for available hardware resources.

3.3.4 Concurrently Parallel Implementation

In addition to the above-mentioned pipeline implementation, the concurrent implementation is also performed to increase the throughput performance of history sensitive process such as functional modules for computing absolute moment and one-bit plane as well as summation of absolute errors (SAE) over a 4x4 block.

Considering 16 pixels within each 4x4 block are independent of each other, so they can be processed in parallel. Since the SIMD-type data packets are adopted in this implementation, eight operators are enough to process 16 pixels concurrently. Moreover, for the sake of minimum number of operators, two data packets with diverse generation identifiers are employed. As a result, only four operators are required to complete the calculation simultaneously rather than eight operators.

The relationship between SIMD-type data packets and image pixel data used in this implementation is illustrated in Fig. 3.9. Fig. 3.9 (a) shows a 4x4 block image. It can be seen that two neighbored pixels are packed into one packet, which are represented as data0 and data1, respectively. Fig. 3.9 (b) illustrates a SIMD-type data packet composed of generation identifiers and double data, and so on. Since current data-driven multimedia processor (DDMP) [58] enables dynamic data-driven processing, the generation identifiers illustrated in Fig. 3.9 (b) (i.e. pixel=12 bits, line=14 bits, frame=2 bits) is required to identify a pair of matched data packets. As shown in Fig. 3.9 (b), pixel (pl) and line (ln) involved in the header of a DDMP data packet denote the horizontal and vertical coordinates of the image, respectively. Otherwise, frame shown in Fig. 3.9 (b) is used to denote the key frame (reference frame) or the frame predicted by the key frame. By virtue of the generation identifiers (i.e. ln and pl), the desired pixel data can be accessed directly from the on-chip associative temporal memory.

Moreover, DDMP empowers many compound instructions related to the associative temporal memory, which significantly contributes to the increased flow rate. Especially, with regard to history sensitive process, desired throughput performance can be accomplished via parallel execution of several identical compound instructions of associative temporal memory.

As an example for this case, the concurrent parallel implementation for computing AM is shown in Fig. 3.10. It can be observed that the AM module receives two mean values with diverse ln values (one is ln=1, another is ln=3) and identical pl values (i.e. pl=2). These two mean values are used to read the original luminance data stored in the associative temporal memory. The mean with ln=1 is responsible for the first two lines over each 4x4 block, otherwise the mean with ln=3 via four identical *read&dif* operators. Next, two add operators are executed concurrently. After that, summation

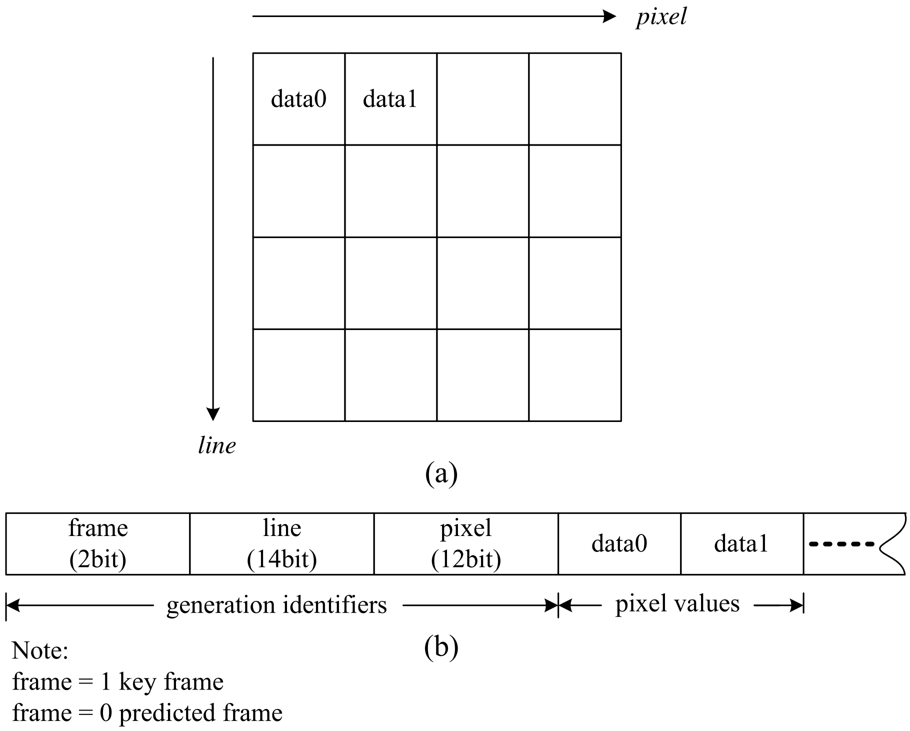


Fig. 3.9 SIMD-type DDMP packet vs. image pixels. (a) A 4x4 block image; (b) A SIMD-type DDMP packet.

of absolute value of the four mean errors is obtained. As the double-data packet is used in this implementation, both data 0 and data 1 contain the summation of four pixels individually. Also, data packets with ln=2i-1 and ln=2i+1 (i=1,3,5......) can be processed simultaneously. Accordingly, two packets with summation of four pixels are produced. To get the summation of 8 pixels, the generation identifiers need to be identical. To this end, the operator used to differentiate them is required. And then the packet with ln=2i+1 goes through the operator for subtracting 2 from ln. Next, the packet holding summation of 8 pixels in data0 and data1 is swapped and added and right shifted 4 bit (due to 16 pixel). Finally, the AM is outputted. It can be derived that the response time of our parallel implementation is around half of that of sequential process due to half length of the critical path.

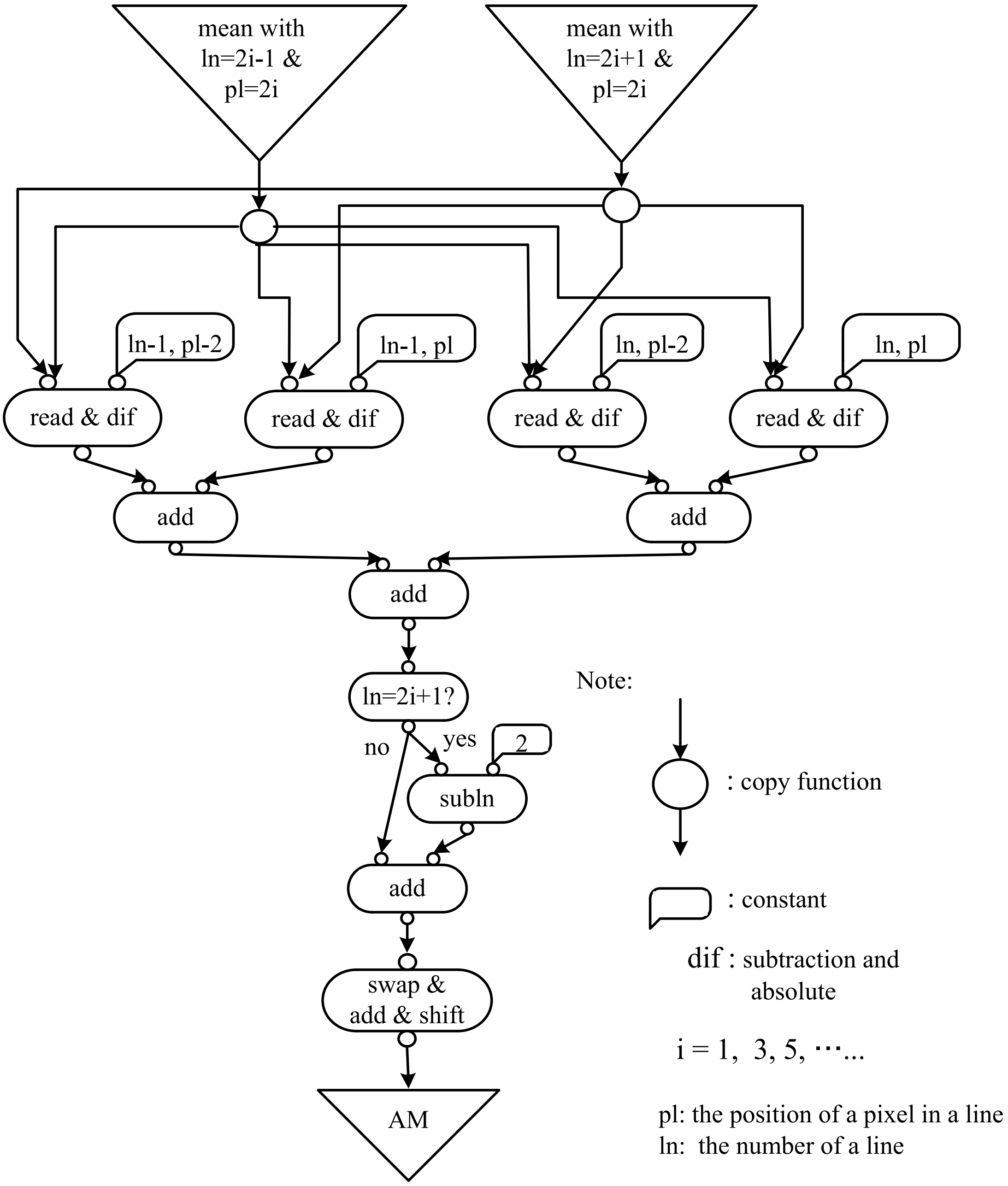


Fig. 3.10 Data-driven parallel implementation of AM module.

3.3.5 Load Balance on Multiprocessor Encoding Systems

(I) Basic principles of load balance

Load balance among the multiprocessor is essential to optimize the performance of a data-driven program. This is because pipeline throughput is slowed down in case of the occupation ratio of pipeline stages over the pipeline efficiency. Pipeline efficiency is the proportion of net processing time spent on packet processing in terms of pipeline throughput [58]. Thus, the load balance on multiprocessor systems is a critical step for the maximum performance of a data-driven program.

In contrast to the conventional processors, it is much easier to build an optimization for individual programs in terms of load balance among the data-driven processors. This is because attentions do not need to be paid to communication overheads and process scheduling [58]. Since a data-driven program is divided into pieces in node-by-node fashion, load balance over the multiprocessor systems can be performed on the basis of resulting pieces. Moreover, the principles for load balance on the data-driven multiprocessor systems are summarized as follows.

Provided that memory resources are used in a software program, then memory allocation is needed to be performed ahead of the load balance. This is because memory allocation determines the allocation of memory relative instructions involved in the instruction sets of current data-driven processors.

The following step is operator identification. Doing this step has two objectives. One is to identify the operators like copy, conditional branch and absorb which tend to change the processing load. Due to the feature of such operators, it is necessary to first extract and place them over the available processors individually. The other is to form various sorts of node pieces according to the category of the operators.

The data-driven chip multiprocessor shown in Fig. 3.11 consists of three types of processors, ALP, LCP and CVP. These processors with different configurations are customized into different tasks. An ALP processor mainly responsible for the arithmetic and logic operations involves a complicated functional module (FP). A LCP processor primarily aiming at the tasks for varying generation identifiers has a simple FP module. A CVP processor with an interface to the external memory is chiefly responsible for

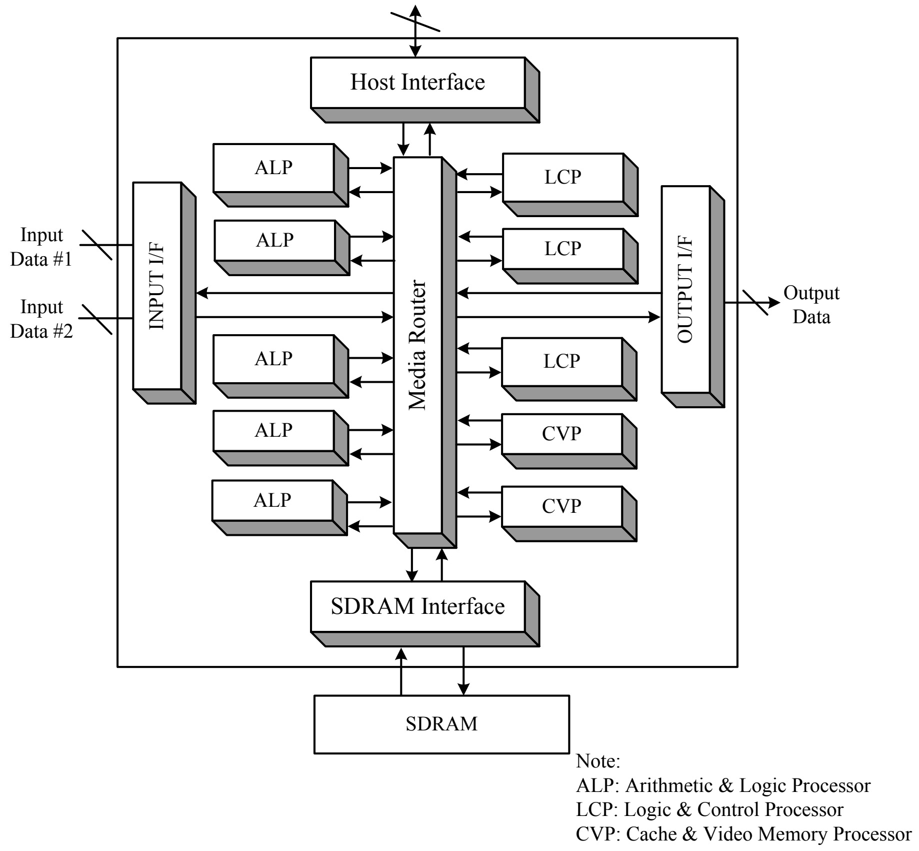


Fig. 3.11 Block diagram of DDMP.

external memory relative operations. Accordingly, the formed node pieces are allocated over the dedicated processors.

It can be derived that no control consideration is necessary. This is because neither context switching nor scheduling is required in the data-driven paradigm. Otherwise, regarding the control-driven paradigm, a complete program debugging on a multiprocessor system is difficult because program scheduling must be considered to ensure that the program sequence is the same as that of the original program. As a consequence, the data-driven processors can benefit more immediately from the today's ever-increasing integrated technologies to fit themselves to multiprocessor systems [58].

(II) Load balance on data-driven multiprocessor systems

It is well-known that load balance among multiprocessor systems is essential for optimum utilization of the component processors. The load balance on four- and eight-processor systems is performed to realize a scalable implementation of intra-frame encoder. As the pipeline stages of CVP is more than that of ALU and LCP, the response time of CVP is greater than that of other processors. Consequently, CVP processors are not used for the history sensitive processes involved in the intra-frame coding such as AM, SAE and SAD. This is because the throughput performance of history sensitive process is dependent on the response time of a critical path.

According to the basic principles of load allocation mentioned above, the memory allocation should be performed at first. The current DDMP shown in Fig. 3.11 is composed of 10 processors on a single chip, each of which has an associative temporal memory with 2048 words (32 bit/word). To reduce the required memory capacity, it is necessary to separate long-length data into short-length ones and compact the scattered data.

In the DPCM module for normal blocks, the number of different pixels between current one-bit plane and its previous one is required. Since the number of bits is 16 for each 4x4 block, the total memory capacity required is equal to 2^{16} words in case of lookup table implementation. While this number is more than the maximum capacity of current associative temporal memory on the DDMP chip. To tackle this problem, a one-bit plane composed of 16 pixels is divided into top 8 bit and low 8 bit, then only 2^8 words of associative temporal memory are demanded. As a consequence, the memory requirement is decreased drastically. And in the same way, the memory management for mean, AM, bit plane and SAE is shown in Fig. 3.12. It can be observed that the scattered moment data belonging to one block are compacted into one block via

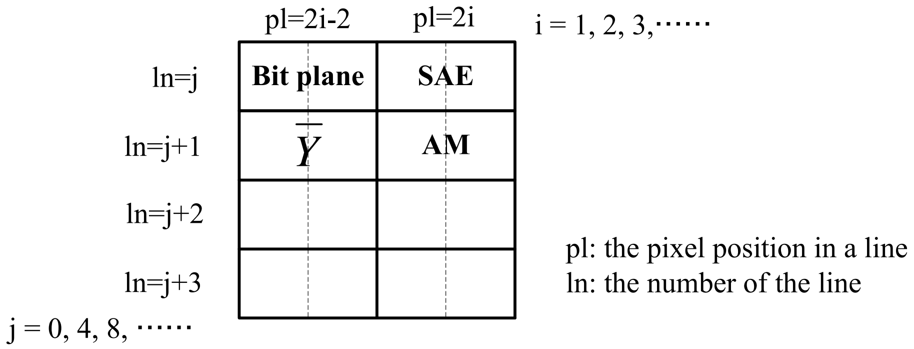


Fig. 3.12 Memory management for the entropy moments of luminance block images.

changing generation identifiers (pl and ln) so that the memory capacity required is reduced too much.

Table 3.1 Memory management for 4 processors.

4 processors	alp0	alp1	alp2	lcp0
data	R G B ME	Y AME	$\overline{Y}$ AM bit plane SAE	Cr Cb $\overline{Cr}$ $\overline{Cb}$ lookup table

The memory management for 4 processors is illustrated in Table 3.1. It can be seen that all three components of a color image, red (R), green (G) and blue (B), are compacted into one processor in order to avoid data overwriting errors in case of concurrent processing. Moreover, all the entropy moments of each luminance block, $\overline{Y}$, AM, bit plane and SAE, are stored into one processor so as to reduce the memory capacity. Considering no relationship between Y component and C component (Cr and Cb), the pixel data of Y component and C component are allocated on different processors to realize concurrent processing. As for the intra-frame coding using 8 processors, the memory management is shown in Table 3.2. The principle of this allocation is identical to that of 4 processors. That is to say, memory allocation is based on the data dependency and amounts of data.

Table 3.2 Memory management for 8 processors.

8 processors	alp0	alp1	alp2	alp3	alp4	lcp0	lcp1	lcp2
data	R ME	B	Y	$\overline{Y}$ AM bit plane SAE	AME	G	Cr Cb $\overline{Cr}$ $\overline{Cb}$	lookup table

In the data-driven program, the elementary data-processing functions are shown by nodes, and data dependency among the nodes is represented by directed arcs connecting the nodes [58]. As for the functional process, the total number of arcs denotes the processing load in the processor. Otherwise, concerning the history sensitive process, processing load is dependent on the data structure residing in the arcs. Accordingly, the processing type should be identified ahead of the load balance.

Since the assignment of the memory relative nodes is determined by the results of the memory allocation mentioned above, it is performed at first. After that, such nodes as copy and conditional branch are allocated due to their feature of varying processing load. Last, the remaining functional nodes are allocated on the suitable processors according to the category of the nodes and the number of arcs so as to make the processing load balance.

The concrete results of load allocation on the four- and eight-processor systems are shown in Table 3.3 and Table 3.4, respectively. In both tables, nodes can be categorized into functional nodes and memory relative nodes. The former contains the arithmetic and logic instructions, conditional branch instructions, copy instructions and compound instructions. On the other hand, the latter involves the memory initialization instructions, memory access instructions and memory compound instructions. It should be noted here that the total number of nodes shown in Table 3.4 is more 4 than that shown in Table 3.3. This is due to the fact that these 4 nodes are utilized to initialize the additional 4 processors involved in the 8-processor encoding system. It can be also observed that the total number of arcs allocated on each processor is not equal because the data dependencies inherent in the ABTC scheme and dedicated instruction sets of each type of processors limit the load balance. Therefore, this is a best allocation result for our data-driven parallel implementation.

Table 3.3 Load balance on 4-processor encoding system.

4 processors	No. of Nodes	No. of Arcs
alp0	65	77
alp1	58	70
alp2	65	80
lcp0	54	74

Table 3.4 Load balance on 8-processor encoding system.

8 processors	No. of Nodes	No. of Arcs
alp0	36	42
alp1	37	48
alp2	26	29
alp3	26	33
alp4	32	35
lcp0	31	40
lcp1	38	50
lcp2	20	28

3.4 Discussion

In this chapter, an adaptive block truncation coding scheme suitable for highly-parallel software implementation was proposed. Moreover, its data-driven parallel implementation was discussed to realize a fast encoding on flexible SoC.

The main points of the proposed ABTC scheme different from the existing BTC-based schemes are shown as follows. The first, threshold values used for three-level classification are different from ACC scheme [28]. In the proposed ABTC scheme, AM and MAE are used within a 4x4 block, while block ranges (i.e. difference of maximum and minimum luminance values) are selected as both threshold values in the ACC scheme. Although block ranges can achieve optimal edge reservation performance, they are incapable of reflecting the variances of luminance over a 4x4 block. The second, three types of blocks are encoded by different ways for adaptive coding based on the characteristics of local image. In the proposed ABTC scheme, to keep more entropy information of complex block images, the mean errors of each pattern block are computed. Moreover, to reduce the bit rate while to produce good reconstruction, the adaptive approach is used to determine the bit number of mean errors. Namely, the bit number is decided by the maximum absolute value of mean errors over each pattern block. Also, in order to simplify the decoding process, the original procedure of AMBTC is simplified without considering the distribution of pixel values over a 4x4 block image. The third, considering adjacent pixels possess high similarity in the identical image, the differential prediction coding modulation (DPCM) is used for intra-frame coding. Furthermore, to reach a better trade-off performance between image quality and computational complexity, three prediction approaches corresponding to block types are applied for intra-frame prediction.

Moreover, an interframe motion prediction approach used in the proposed ABTC scheme is a simple block-based difference coding in that a current block is only compared with its counterpart in the previous frame. This prediction method is also more compatible with the AMBTC-based intra-frame coding than motion estimation widely used in the current video coding standards such as H.263+/H.264. This is because the entropy moments of each block image produced by AMBTC are used as arbitrary metrics. Again, motion estimation is a time-consuming process although it achieves higher motion prediction efficiency.

Furthermore, its data-driven parallel software implementation was performed on the self-timed super-pipelined multiprocessor. For maximum firing rate of the pipeline circuit, the parallelism inherent in the proposed ABTC scheme are exploited based on the data dependencies. Given that there are data dependencies between elements, then the pipelined parallelism can be exploited. Otherwise, concurrent parallelism.

Since the response time of a critical path in a history sensitive process impacts on the throughput performance, the length of the critical path must be minimized. To this end, the compound instruction and associative temporal memory as well as the SIMD-type data packets are taken full advantage of in this software implementation. Two examples, Mean module and AM module, were illustrated in this chapter.

In addition, data-driven processors enable the latency-tolerant execution. This is derived from its firing rule that any operation is initiated only when a couple of data necessary for this operation become available. Thus the firing rate is not hindered by the delay time produced by the memory access and processing delay as well as inter-processor communication. However, in the conventional processor, where the instructions are executed in a predetermined sequence, such kind of latencies residing in the processors affects the execution of sequential instructions [58].

To keep the maximum flow rate of each processor in a chip-multiprocessor system, load balance is necessary. This is because pipeline throughput is affected by the occupation ratio of pipeline stages. Provided that the occupation ratio is over pipeline efficiency, the flow rate of packets is cut down. Fortunately, load balance among data-driven processors is rather easier than doing that among control-driven processors because of no control considerations.

In order to realize the load balance, the memory allocation should be performed at first in that it determines the allocation of memory relative instructions. Moreover, to minimize the required memory capacity, each 16-bit map data used to reserve the local features is separated into top 8-bit and low 8-bit data. Then two 8-bit data of a current block are compared with that of its previous one, respectively. In this case, the size of lookup table becomes small so that memory requirement is reduced too much. For the identical purpose, the scattered entropy moment data of block images are managed into one processor based on the tagged information for the identical purpose.

Since such functional nodes as copy, conditional branch and absorb alter the processing load, it is necessary to extract and place them among the available processors individually. After that, the remaining functional instructions are allocated on the appropriate processors based on the type of instructions and the number of arcs so as to make the processing load balance.

The performance of the proposed ABTC scheme implemented on the dynamic data-driven chip-multiprocessor will be evaluated and a series of experimental results will be illustrated and discussed in the following chapter.

Chapter 4

Experimental Evaluation

4.1 Introduction

In order to verify the performance of the proposed ABTC scheme implemented on the data-driven processing system, experimental evaluation is performed using a variety of benchmark images and video sequences. Rate-distortion performance is measured by the bit rate with the desired image quality or image quality with the target bit rate. Image quality is evaluated from both subjective and objective quality measurements. Subjective quality is illustrated by the decoded image, while objective quality is measured by peak signal to noise ratio (PSNR). Frame rate is referred to as number of encoding frames per second.

In this chapter, the image quality and bit rate of the proposed ABTC scheme is first evaluated from intra- and inter-frame coding. After that, the frame rate and response time of data-driven implementation are evaluated from intra- and inter-frame coding. Moreover, the performance comparison is conducted with the coding standards and similar coding schemes. Finally, a discussion is given at the end of this chapter.

4.2 Image Quality and Bit Rate

Image quality and bit rate are important metrics for measurement of rate-distortion performance. In this section, the decoded image quality with desired bit rates is evaluated from intra- and inter-frame coding. Experimental evaluation is conducted using the benchmark images shown in Fig. 4.1 and Fig. 4.2 and video sequences shown in Fig. 4.8. Moreover, performance comparisons are made for image quality at the same bit rate.

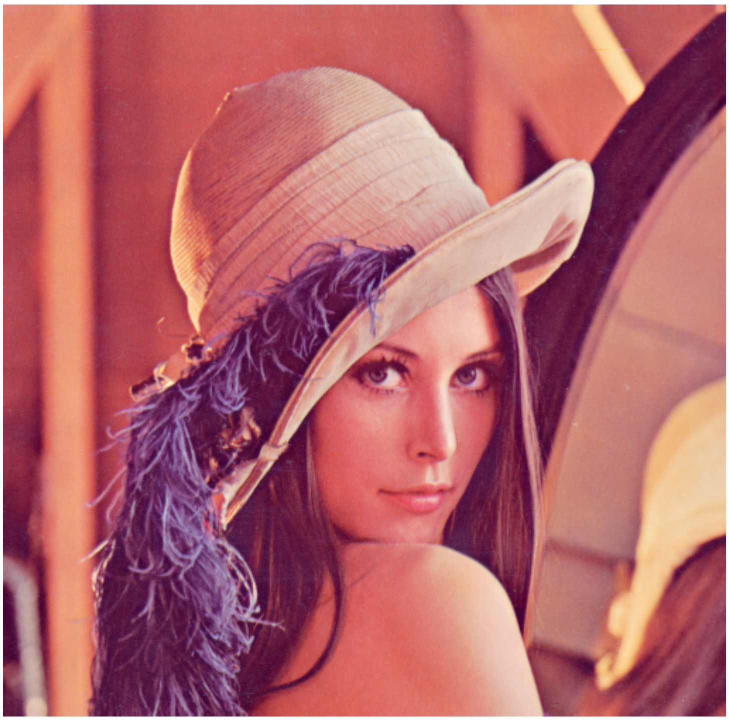
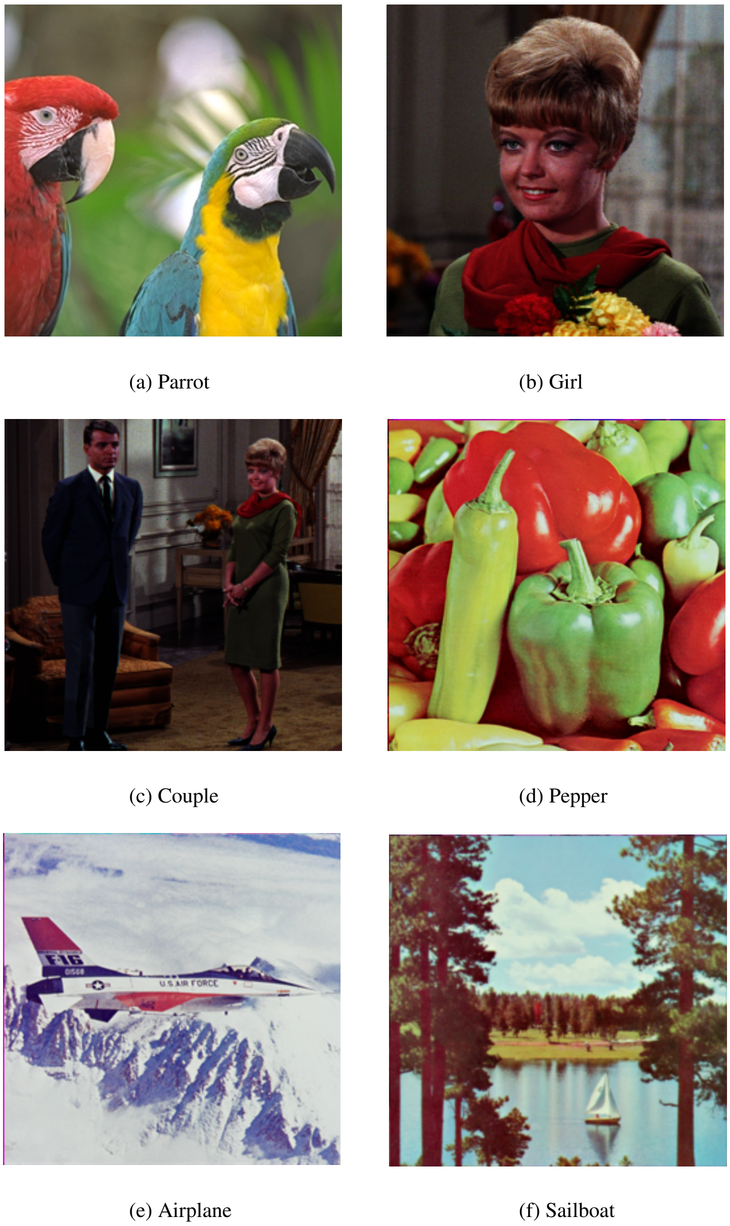


Fig. 4.1 Lena with the size of 512x512.

4.2.1 Intra-Frame Coding

(I) ABTC algorithm

Experimental evaluations were performed using a series of different benchmark images. Original image of Lena with the size of 512x512 is shown in Fig. 4.1. Except for Lena, the size of other images shown in Fig. 4.2 is 256x256. The results shown in Fig. 4.3 indicate that the proposed adaptive BTC (ABTC) achieves image quality around 3 dB better than AMBTC with the 2 bits per pixel [69].



(a) Parrot

(b) Girl

(c) Couple

(d) Pepper

(e) Airplane

(f) Sailboat

Fig. 4.2 Benchmark images with the size of 256x256.

Moreover, the visual quality of the proposed ABTC is tested and compared to other image coding algorithms using Lena image. In order to clarify the difference of visual quality, a part of Lena image reconstructed by four algorithms are enlarged 3 times and presented in Fig. 4.4. It is apparent that the visual quality of the proposed ABTC algorithm is much better than that of both AMBTC and BTC in exchange for little

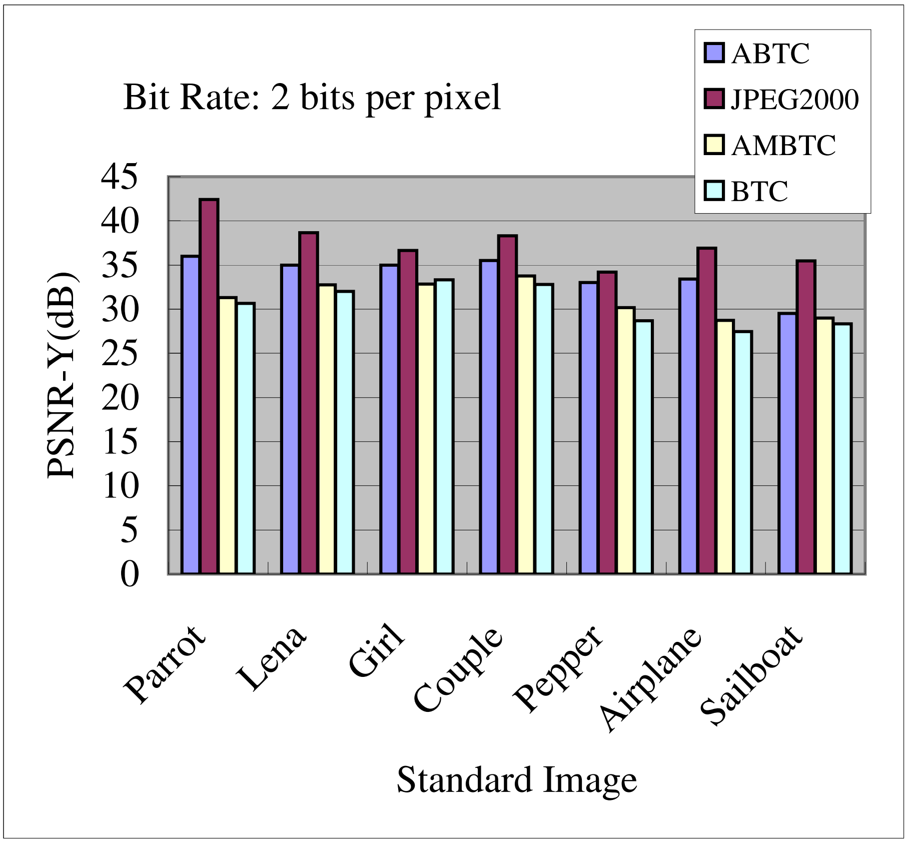
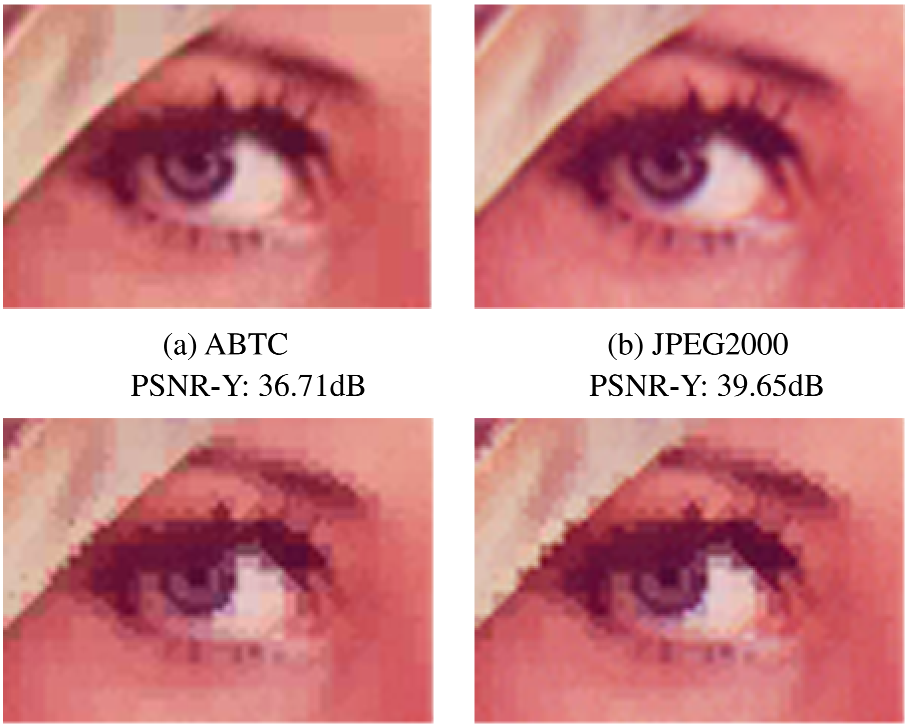


Fig. 4.3 The comparison of image quality.

increase of computation cost. Furthermore, visual quality of the proposed algorithm is competitive to that of JPEG2000 while its computational complexity is much less than that of JPEG2000.

(II) ABTC algorithm coupled with SRQ

The proposed SRQ [63] is aimed to improve the bit rate performance of ABTC algorithm. In order to evaluate the coding gains of ABTC algorithm coupled with SRQ, the performance comparison between the ABTC algorithm and ABTC coupled with SRQ is conducted using the benchmark images. To examine the image quality against compression ratio, the threshold values of AM and SAE are changed for desired



(a) ABTC
PSNR-Y: 36.71dB

(b) JPEG2000
PSNR-Y: 39.65dB

(c) AMBTC
PSNR-Y: 33.02dB

(d) BTC
PSNR-Y: 32.53dB

Fig. 4.4 The comparison of visual quality of reproduced images with the 3 bits per pixel.

compression ratio while reserving the good reconstructed images.

With identical threshold values of AM and SAE, the results of Lena and Airplane as well as Pepper are shown in Fig. 4.5, Fig. 4.6 and Fig. 4.7, respectively. It can be seen that the ABTC coupled with SRQ can produce better image quality than ABTC with identical bit rate at the cost of increased computation cost. While it should be noted that the impact of SRQ on rate-distortion performance of ABTC is varied with the number of pattern blocks. That is to say, with the increased compression ratio, SRQ will not significantly impact on the image quality and bit rate of ABTC.

Since these experiments are targeted for the bit rate gains with the reasonable image quality, the PSNR values of decoded images shown in Fig. 4.5, Fig. 4.6 and Fig. 4.7 are all more than 35 dB, in that over 35 dB of image quality usually gives good visual

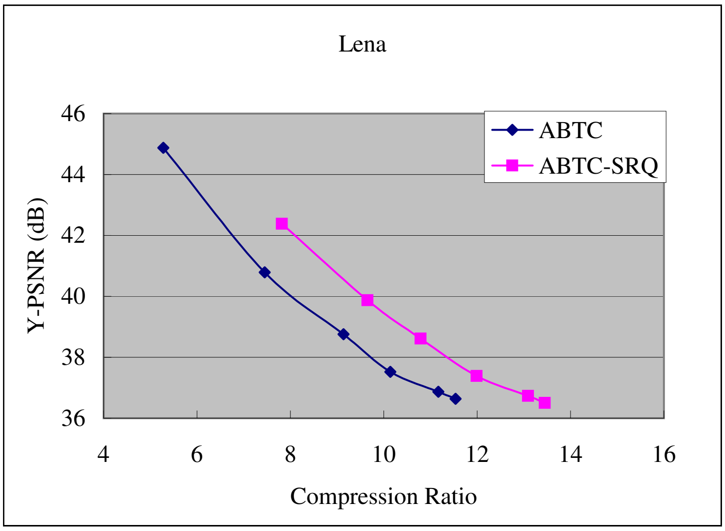


Fig. 4.5 The performance comparison between ABTC and ABTC with SRQ using Lena.

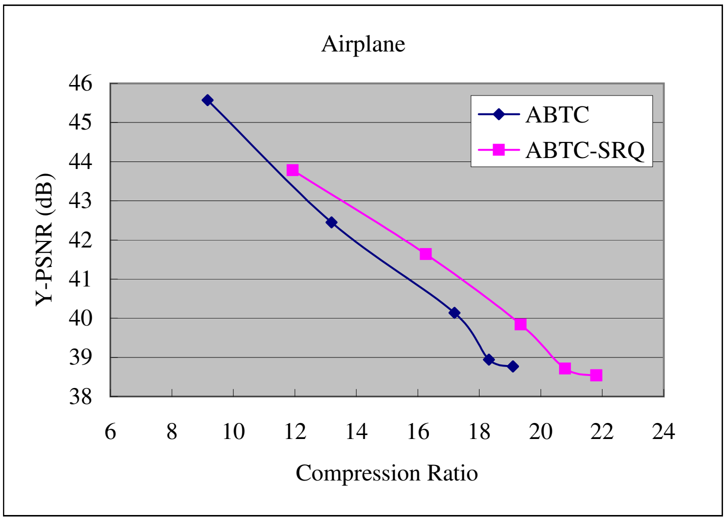


Fig. 4.6 The performance comparison between ABTC and ABTC with SRQ using Airplane.

perception. While the compression ratio is not large so that the number of pattern blocks is large. In this case, the effect of SRQ on coding efficiency of original ABTC is significant. Otherwise, in case of large compression ratio, the number of pattern blocks is so small that the coding gains of ABTC coupled with SRQ can be negligible. This is the reason why ABTC algorithm is solely applied to the interframe coding discussed in the following subsection.

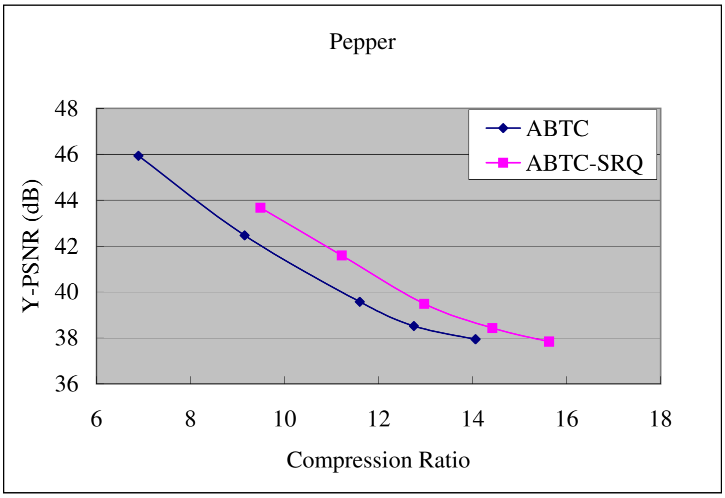


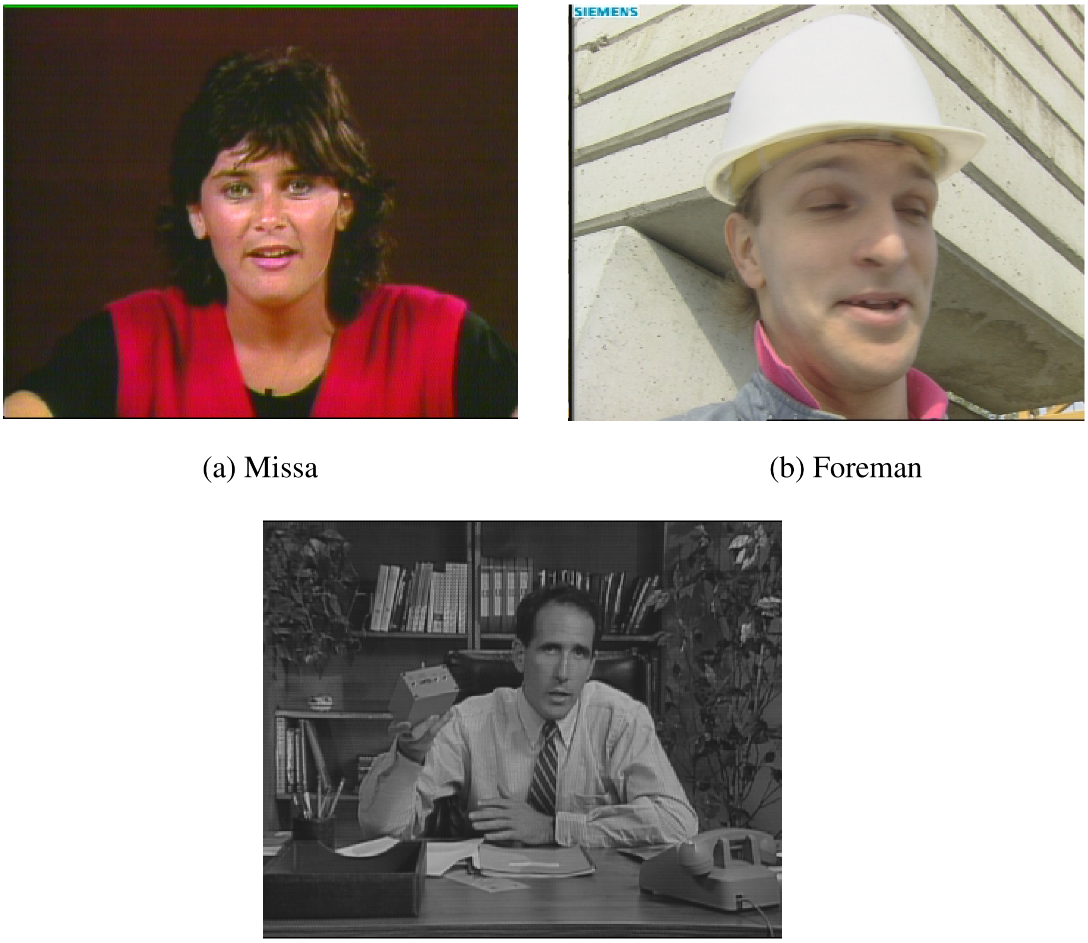
Fig. 4.7 The performance comparison between ABTC and ABTC with SRQ using Pepper.

4.2.2 Inter-Frame Coding

In this subsection, the performance evaluation of the proposed ABTC scheme including ABTC algorithm and BTC-based motion prediction approach is performed using standard video sequences with different motion activities and background images. The third frame of each video sequence is shown in Fig. 4.8. The characteristics of these video sequences are shown in Table 4.1. Moreover, with the identical threshold values, the features of test video images were tested and summarized in Table 4.2. It is obvious to observe that video sequences tested feature different block images; especially Miss America (Missa) possesses the percentage of blocks same to its previous one up to 42.1% so that using DPCM can reduce the intra-frame redundancies very much.

Table 4.1 Test video sequences. (Size: CIF; No. of tested frames: 30.)

Video sequence	Missa	Foreman	Salesman
Motion	Slow motion	Large motion in all directions with camera motion	Fast motion
Information	Scanty	Scanty	Noisy background



(a) Missa

(b) Foreman

(c) Salesman

Fig. 4.8 The 3rd frame images of original video sequences.

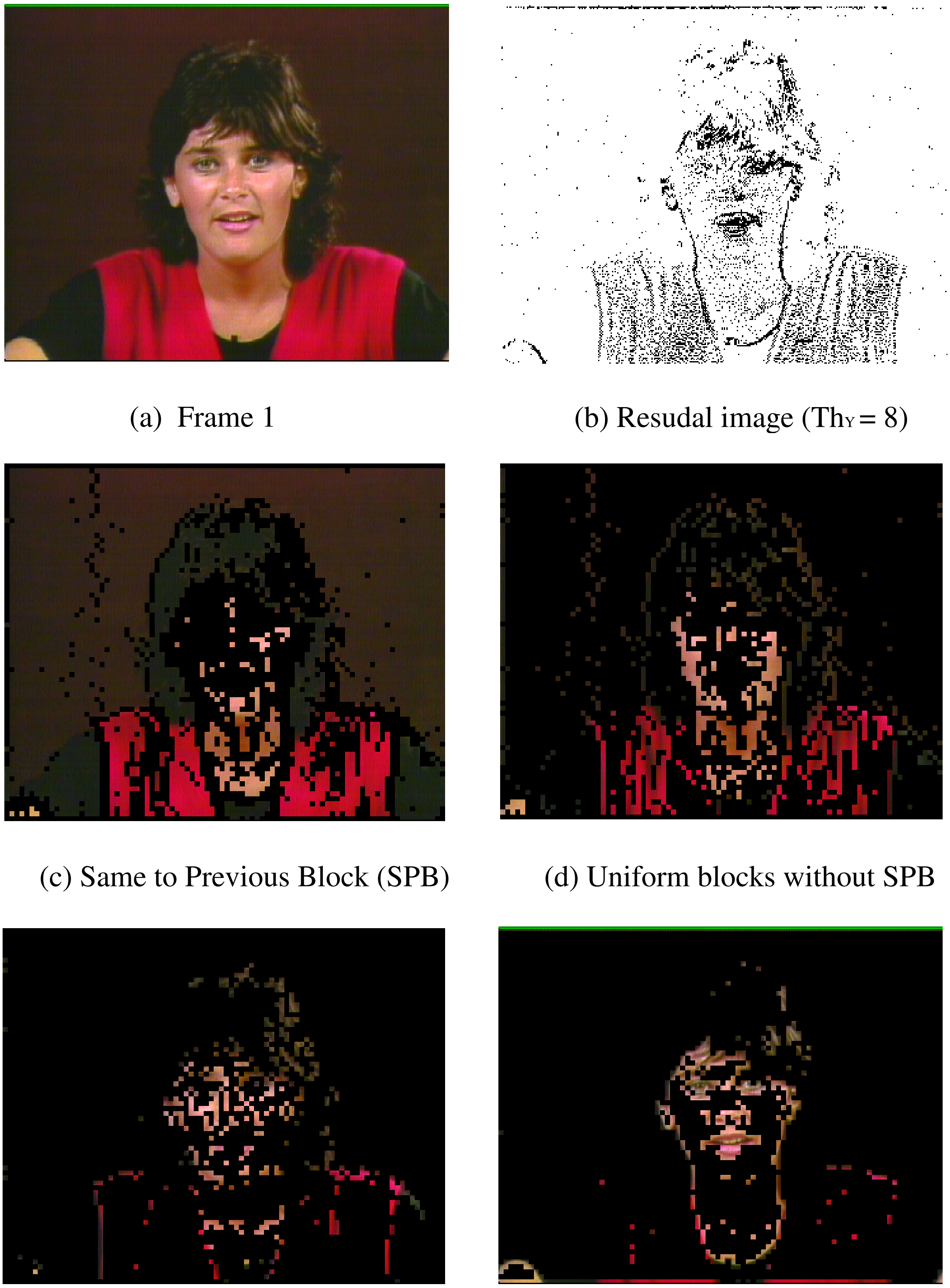
The residual image and predefined block images of Frame 1 of Missa video sequence were analyzed. Then the resulting images were illustrated in Fig. 4.9. A residual image shown in Fig. 4.9 (b) was produced by subtracting Frame 0 from Frame 1 without motion prediction with the threshold of luminance difference equal to 8. Generally, the human vision system (HVS) is not sensitive to the luminance variance less than 8. It can be observed from Fig. 4.9 (b) that the large redundant information exist in the consecutive frames. This provides the possibility for the coding gains by inter-frame prediction method. Moreover, the various block images in Frame 1 were illustrated in Fig. 4.9 (c), Fig. 4.9 (d), Fig. 4.9 (e) and Fig. 4.9 (f), respectively. It can be observed

Table 4.2 Characteristics of block images in the tested video sequences.

Block type		Uniform	Normal	Pattern	SPB
Missa	Amount	781	1270	1620	2665
	Percentage (%)	12.3	20.0	25.6	42.1
Foreman	Amount	1003	1532	2771	1030
	Percentage(%)	15.8	24.2	43.7	16.3
Salesman	Amount	493	1053	4237	553
	Percentage (%)	7.8	16.6	66.9	8.7

from Fig. 4.9 (c) that the number of same to previous block under the predefined threshold values is large. This indicates that the intra-frame prediction approach is necessary to remove the redundancies within a head-shoulder image like Missa. In these experiments, compression ratio, peak signal to noise ratio (PSNR) are selected as quantitative evaluation measurements.

All the sequences are colored images with 24 bit per pel and the first 30 frames of each sequence were tested. The size of these frame images are $288 lines \times 352 pixels$ (CIF). Since the first evaluation is aimed at testing the rate-distortion performance of the proposed ABTC scheme, the threshold values were varied to examine sensitivity of PSNR against compression ratio. In order to obtain desired compression ratio, the optimal combinations of variant threshold values shown in Table 4.3 were investigated using the program. The other experimental conditions constitute 8 bits for each mean of luminance component and 5 bits for each AM as well as 6 bits for each mean of chrominance components. Moreover, the threshold value of *difMean* and *difAM* is equal to that of AM. In addition, the threshold value of *difMap* is 5. The experimental results are provided in Fig. 4.10. It can be seen that the proposed scheme can achieve very good rate-distortion performance for Missa due to low bit rate of 0.4 bit/pel and



(a) Frame 1

(b) Resudal image ($\mathrm{Th}_{Y} = 8$)

(c) Same to Previous Block (SPB)

(d) Uniform blocks without SPB

(e) Normal blocks without SPB

(f) Pattern blocks without SPB

Fig. 4.9 The various block images in Frame 1.

PSNR of around 37 dB (decibels). The reconstructed images for all third frames of test video sequences are shown in Fig. 4.11. It is apparent to observe that reconstructed image of Missa produced is better than that of Foreman or Salesman within the tested compression ratio due to its little changes between successive frames. These results indicate that the proposed ABTC scheme is more suitable for semi-motion pictures with scanty information and slow motion.

Table 4.3 The parameters used in the experiment.

Desired CR		10	15	20	25	30	35	40	45	50	55	60
Missa	Th_{AM}	2	2	3	4	4	4	5	5	5	5	5
	Th_{SAE}	32	46	50	57	60	63	73	76	80	81	85
	Cut-error	0	1	1	2	2	2	2	2	2	2	2
Foreman	Th_{AM}	3	4	5	6	7	8	9	10	12	13	13
	Th_{SAE}	50	68	90	115	130	147	167	177	186	200	208
	Cut-error	1	2	2	2	3	3	3	3	4	4	4
Salesman	Th_{AM}	3	4	4	5	5	5	5	6	7	7	8
	Th_{SAE}	30	50	68	80	88	92	98	108	118	128	137
	Cut-error	1	2	2	2	2	3	3	3	3	3	3

4.3 Frame Rate and Response Time

The second evaluation is targeted at testing the frame rate and response time of the ABTC scheme implemented on the chip-multiprocessor encoding system. The frame rate was measured by maximum throughput denoted as MaxTP. MaxTP refers to the maximum number of double-data packets per second. Frame rate (FR) is calculated by equation (4.1). In Table 4.5, response time (RT) is the duration time between the input time of the first pair of pixel data and its output time.

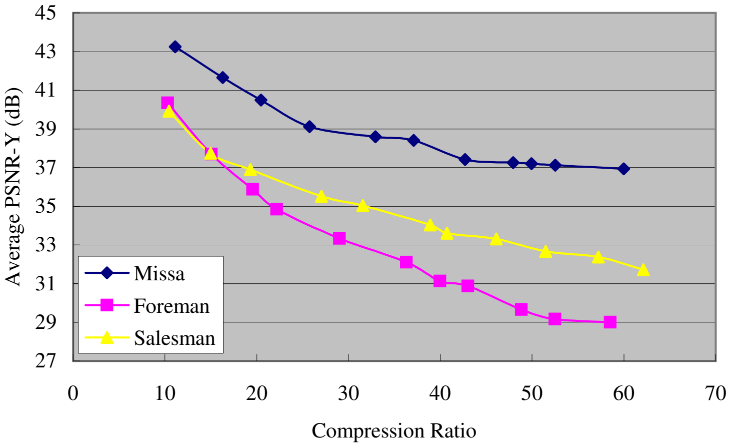
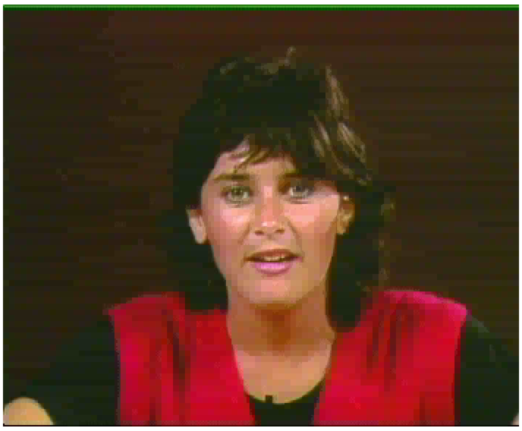
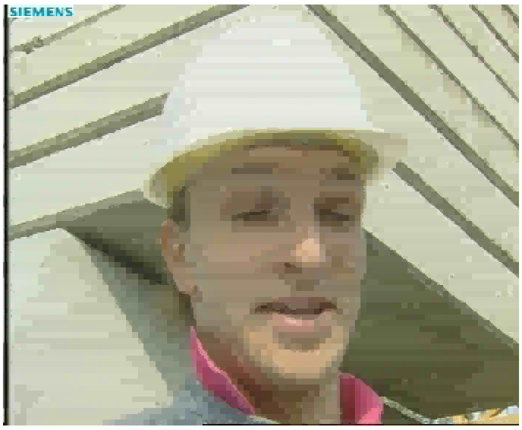


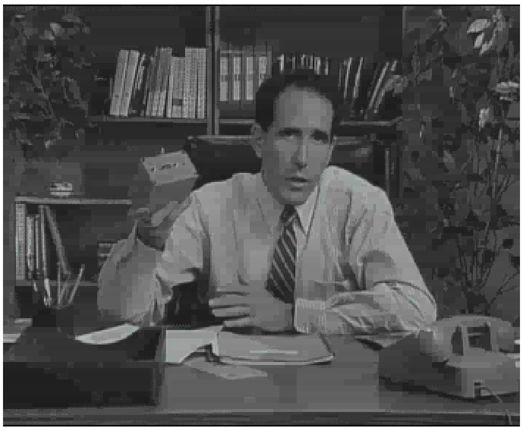
Fig. 4.10 The comparison of rate-distortion performance.



The 3rd Frame of Missa



The 3rd Frame of Foreman



The 3rd Frame of Salesman

Fig. 4.11 Comparison of visual quality of reconstructed images with the 60 compression ratio.

$$FR = \frac{MaxTP}{PictureSize/2} \tag{4.1}$$

4.3.1 Intra-Frame Coding

Concerning intra-frame coding, the data-driven parallel implementation of the proposed ABTC scheme was performed using variant number of processors on a single chip. With the identical parameters, the result shown in Table 4.4 illustrates that over 30 VGA images per second were achieved using four processors. Moreover, as for an ideal case, the performance of data-driven software implementation should be scalable along with the increase in number of available processors. While as for this practical implementation, it is difficult to achieve the proportional speedup since the essential data dependencies inherent in the ABTC algorithm limit the load balance on available processors. The best performance has therefore been achieved in this data-driven parallel implementation. This is due to the fact that the speedup ratio of frame rate shown in Table 4.4 is equal to around 1.6 in the average case.

Table 4.4 Frame rate of ABTC encoder using VGA image.

Metrics	Best	Typical	Worst	Average
Block type	Uniform	Normal	Pattern	Combined
4 processors	54fps	30fps	28fps	37fps
8 processors	77fps	54fps	48fps	59fps

4.3.2 Inter-Frame Coding

Regarding inter-frame coding, a performance comparison between the scanline-order and block-order input was conducted. The results are shown in Table 4.5. It can be deduced that throughput performance obtained by block-order input is around twofold as many as that of scanline-order input on average. It can also be observed that response time of the block-order input is not affected by image size, while that of scanline-order input is incremental along with the increase of image size. Hence a block-order input for the proposed ABTC scheme is superior to the original scanline-order input. These results shown in Table 4.5 are derived from a special video sequence where no block image is identical to its previous one in both previous frame and identical frame. Thus the frame rate can be further improved for the practical images.

Table 4.5 Performance comparison between scanline- and block-order input for inter-frame coding using 10 processors on a chip.

Size	Input	Metrics	Best	Typical	Worst	Average
		Block	Uniform	Normal	Pattern	Combined
VGA	Block	FR (f/sec)	99	69	61	76
		RT (us)	25.6	33.4	36.3	31.8
	Scanline	FR (f/sec)	48	24	20	30
		RT (us)	186.4	348.4	425.2	320
CIF	Block	FR (f/sec)	300	212	190	234
		RT (us)	25.6	33.4	36.3	31.8
	Scanline	FR (f/sec)	150	87	66	101
		RT (us)	113.8	184.5	235.0	177.7

4.4 Performance Comparison

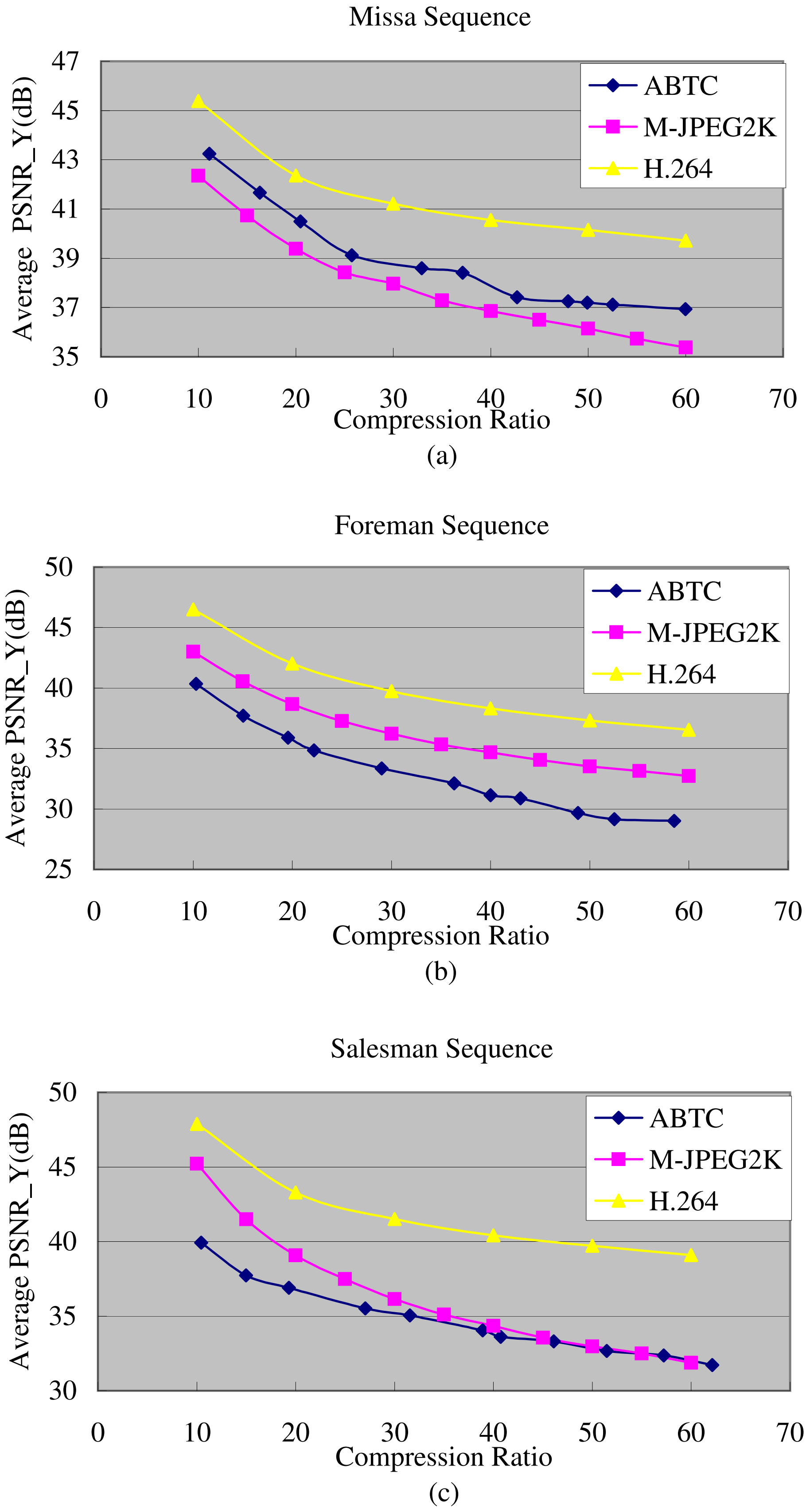
The performance comparison was performed in two steps using three video sequences. The first step is to compare with compression standards, Motion-JPEG2000 by virtue of jj2000-4.1 [70] and H.264 (baseline profile) using JM9.4 [71]. In the second step, the ABTC scheme is compared with other modified BTC scheme [22]. The comparison made here focuses on trade-off performance between reconstructed quality and computational complexity as well as memory complexity. The results are summarized in Table 4.6.

Table 4.6 Performance comparison of different coding algorithms.

		ABTC2	ABTC1 [22]	Motion JPEG2K	H.264
Intraframe		ABTC2	ABTC1	DWT	4x4 integer DCT
Interframe		BTC	BTC	None	MC
Average PSNR(dB)	Missa	37	35.7	35	40
	Foreman	29	27.8	33	37
	Salesman	32	32	32	39
Bpp (bit/pel)		0.4	0.4	0.4	0.4
Computational Complexity for CIF image at 30 fps (MOPS)		18.8	12	1997.4	2081.8

(I) PSNR

The comparison of rate-distortion performance was performed and shown in Fig. 4.12. It can be recognized from Fig. 4.12 (a) that the average luminance PSNR values for Missa using Motion-JPEG2000 and H.264 are around 35 dB and 40 dB at the bit rate of 0.4 bit/pel, respectively. Moreover, the average luminance PSNR value using the proposed ABTC scheme is around 37 dB at the same bit rate. Since all luminance PSNR values are over 35 dB, ABTC scheme can be addressed as same level as other



(a)

(b)

(c)

Fig. 4.12 Comparison of rate-distortion performance.

high-quality coding standards. On the other hand, with the large-motion and camera-motion or fast-motion video sequence such as Foreman or Salesman, the reconstructed quality of the ABTC scheme is not as good as that of the standard algorithms under the large compression ratio, in that a large amount of high-detail pattern blocks are identified as uniform blocks or normal blocks encoded by mean or AMBTC.

(II) Computational complexity

Suppose here that addition, subtraction, multiplication, logic operation are considered as individual computational entities. In H.264, the hierarchical integer DCT is employed instead of the original DCT used in the H.263. The size of these transforms is mainly 4x4, in special cases 2x2 [7]. To simplify the quantization of computational complexity, the amount of computation of H.264 shown in Table 4.5 was obtained by counting up the operations of 4x4 integer DCT and MC. While H.264 endures more computational complexity than its predecessors such as MPEG2/MPEG4 due to new encoding tools such as variable block sizes for ME/MC and CAVLC entropy coding in case of baseline profile. In Motion-JPEG2000, discrete wavelet transform (DWT) is used to transform the spatial domain to temporal frequency, which is a computation-intensive operation. It requires the computation cost up to 6.66×10^7 for the CIF image. Otherwise, concerning the ABTC scheme, the computational requirements are around 6×10^5 for intra-frame coding and approximately 1.9×10^4 for intra-frame DPCM in case that all the blocks are identified as pattern blocks.

Motion compensation (MC) in case of full search (FS) requires $3(2p + 1)^2 MN$ operations, where $p(= 7)$ is the range of search and MN is the size of image. The BTC-based motion prediction approach requires $3n$ operations in case of pattern blocks, where n is the number of pixel blocks.

(III) Memory complexity

In the proposed scheme, one sixteenth of each frame is required to be stored into memory, but in the baseline profile of H.264 one whole frame is required to be deposited for motion estimation. Furthermore, memory access operations of the proposed ABTC scheme can be conducted in parallel to the other computations on a block-by-block basis, so that memory access latency can not affect total throughput.

As for the comparison to AMBTC-based scheme, it can be derived from Table 4.6 that the proposed ABTC can produce around 1.3 dB better reconstructed quality than the previous ABTC scheme [22] for Missa at the identical bit rate with little increase of computation cost. Accordingly, the proposed scheme achieves a better trade-off between reconstructed quality and computational complexity.

4.5 Discussion

In this chapter, the proposed ABTC scheme on the data-driven chip multiprocessor was evaluated in terms of rate-distortion performance and frame rate as well as the response time. Moreover, the performance evaluation was performed for intra-frame and inter-frame coding by virtue of various benchmark images and standard video sequences.

For the intra-frame coding, the ABTC algorithm was compared with the similar algorithms (i.e. BTC and AMBTC) and coding standard JPEG2000. The experimental results show that the ABTC can produce better image quality than BTC and AMBTC with the identical bit rate. While the image quality of the proposed ABTC is not good as that of JPEG2000, its computational complexity is much less than that of JPEG2000. A heavy coding standard like JPEG2000 requires a high-speed processor for realizing the real-time applications. Especially, users today run the multiple demanding applications at the same time, so that a more powerful processing platform is required. In order to meet this requirement, clock frequency of a single programmable processor is continually increased. However, a very high clock speed tends to result in the clock skew problem and high power consumption. Moreover, today's hardware engineers have recognized that it is hard to further improve the performance of a single programmable processor by increasing the clock frequency in the future.

As a consequence, the multi-core processors have emerged for meeting the multi-function applications so as to overcome the clock skew problem and diminish the power dissipation. For example, Intel. Corp [62] has launched the latest dual-core processor named as Pentium-D which offers the exceptional functionality and performance. However, the multi-core Von Neumann-type processor will bring the crisis to the software design. This is because both process scheduling and inter-processor communication are needed to be considered by the software engineers which results in the long design circle. On the other hand, the data-driven software implementation for the multi-core processor is much easier owing to no consideration of the process scheduling and inter-processor communication. Additionally, data-driven computing paradigm allows us to represent system functions in natural way, including hardware and software portion seamlessly. This advantage leads to improve the SoC design productivity as well as to explore many parallel algorithms easily.

Furthermore, the performance comparison was conducted between the proposed ABTC algorithm and the ABTC coupled with SRQ presented in [63]. The experimental result shows that the ABTC coupled with SRQ can obtain better rate-distortion performance than that of ABTC at the cost of the increased computation cost. While it should be noted here that the the impact of SRQ on the coding efficiency of ABTC becomes smaller along with the increased the compression ratio. This is because the number of pattern blocks is small with the large threshold of SAE. It can be derived that the coding efficiency of ABTC coupled with SRQ is similar to that of ABTC in case of the large compression ratio. Accordingly, with respect to the inter-frame coding, the SRQ is not applied for coding pattern blocks.

Combined with a simple BTC-based motion prediction approach, the ABTC scheme was evaluated utilizing video sequences including diverse background images and motion activities. The experimental results illustrate that the proposed ABTC can achieve reconstructed images up to around 37 dB with 60 compression ratio on average for the Missa, which is better than the other video sequences (i.e. Foreman and Salesman). This result also indicates that the proposed ABTC scheme is more suitable for the semi-motion pictures observed in the applications such as person-to-person teleconferencing and remote surveillance.

Compared to the previous scheme [22], around 1.3 dB image quality gains were accomplished with the identical bit rate. Moreover, the proposed scheme possesses better adaptability than its predecessor owing to its deeper classification and DPCM. Furthermore, the proposed ABTC was compared with the coding standards (i.e. Motion-JPEG2000 and H.264 (baseline)). The experimental result shows that the proposed ABTC scheme can produce better image quality than Motion-JPEG2000, whereas not as good as H.264. Nevertheless, the latest coding standard H.264 endures more computational complexity than the proposed ABTC scheme. As a consequence, the proposed scheme can achieve a better trade-off between the image quality and computational complexity.

The data-driven parallel implementation of the proposed ABTC scheme on data-driven multimedia processor (DDMP) [58] was evaluated in terms of the frame rate and response time. The results of the scalable implementation indicate that over 30 VGA frames per second was achieved using 4 processors. Moreover, the encoding speedup ratio up to 1.6 was obtained by 8 processors. Although the encoding speedup should be scalable along with the increased number of processors for data-driven processing system, it is difficult to achieve for the practical implementation. This is because data dependencies inherent in the algorithm limit the load balance. Accordingly, it can be derived that the best throughput performance has been accomplished.

In addition, a performance comparison between scanline- and block-order input was conducted for inter-frame coding. The results indicated that block-order input produced the superior frame rate performance to the scanline-order because of no three-line delay time. While additional functional module is required to transform the original scanline-order input into the 4x4 block images. To this end, a compound reference instruction of line buffer was proposed. Moreover, the experimental result shows that the response time of block-order input is not varied with the increased size of the image. On the other hand, for scanline-order input, larger size image is, longer delay time becomes. So that the block-order input is superior to scanline-order input for the proposed ABTC scheme. Furthermore, the evaluation result of data-driven parallel implementation shows that around 60 VGA frames per second, which is around twofold as many as that of previous frame [22].

Chapter 5

Conclusion and Future Directions

5.1 Conclusion

In this thesis, an adaptive block truncation coding scheme suitable for highly-parallel software implementation was proposed for semi-motion pictures observed in the applications such as person-to-person teleconferencing and remote surveillance. Moreover, its data-driven highly-parallel software implementation was discussed to realize a fast coding on flexible SoC. The extension of this work towards algorithm and implementation is depicted. Finally, future directions for this study are outlined at the end of this chapter.

5.1.1 An Adaptive Block Truncation Coding Scheme

An adaptive block truncation coding scheme was proposed to realize a fast coding on SoC. The proposed scheme is one of extension of basic BTC so that it is very simple. In order to improve the coding efficiency of basic BTC, three techniques were employed in the proposed scheme. The first is two optimal threshold values for realizing the adaptive coding, the sample first absolute central moment (AM) and mean of absolute errors (MAE). AM represents the dispersion from the mean over a 4x4 block image, it is thus used to differentiate the smooth regions. By virtue of AM, the block images are grouped into two categories, named as uniform block and normal block. Provided that AM is smaller than the predefined threshold, the block is identified as uniform block, otherwise, normal block. In order to differentiate the blocks with large image distortion from the normal blocks, the second threshold MAE is employed as it denotes the errors between the decoded block and its original one. Compared with the range thresholds used in ACC [28] which are targeted at edge reservation, the luminance variance over a block image is considered in the proposed classification approach. Suppose the MAE of a normal block is greater than the preset threshold value, the block is named as pattern block.

The second is variant coding approaches corresponding to three types of block images. To realize a better trade-off between the image quality and computational complexity, each type of block image is encoded by the approach adaptive to its image characteristic. Concerning a uniform block, the mean value of this block is merely computed, while the normal block is encoded by AMBTC. Otherwise, the mean error coding method is utilized to keep more entropy information of pattern blocks.

The third is differential pulse coding modulation (DPCM). To improve the coding efficiency, the DPCM is used to remove the redundancies inherent in the consecutive frames and neighbor pixels. In the inter-frame DPCM, two prediction approaches are utilized for uniform block and normal block. In fact, inter-frame DPCM is a simple block-based difference coding because the current block is only compared to its counterpart in the previous frame. While with respect to the intra-frame DPCM, three adaptive approaches are used to determine the block same as its previous one.

The experimental results of intra- and inter-frame coding have proved that the proposed ABTC scheme can achieve a better trade-off performance between the image quality and computational complexity than its similar algorithms or coding standards. In addition, the proposed scheme features bit rate scalability by means of adaptive threshold values. Thus it can fulfill various user requirements based on available bandwidth and end-system capability. From the evaluation result, around 37 dB image quality was accomplished for Missa video sequence with the 0.4 bits/pel. Moreover, the experimental evaluation indicates that the proposed scheme is more suitable for semi-motion pictures including smooth background image and slow motion like Missa. However, its performance regarding motion-intensive or camera motion video (e.g., Foreman and Scalesman) is degraded. There is no prediction gain for interframe, since these kind of video sequences result in a small number of Same-to-Previous-Frame (SPF) blocks. This is because the interframe prediction approach used in the proposed ABTC scheme only compares the current block to its counterpart in the previous frame. Moreover, the proposed approach requires the additional bits to denote the type of blocks. Consequently, there is less advantage when the proposed scheme is applied for motion-intensive video sequences.

5.1.2 The Data-Driven Parallel Implementation

As for its data-driven implementation, both pipeline parallelism and concurrency were exploited to realize a fast encoding on data-driven chip-multiprocessor. To achieve a desired throughput performance required by the video coding applications, three techniques were adopted in this implementation.

The first is the SIMD-type data. That is, two adjacent pixels are packed into one data packet so as to obtain the data-level parallel process. The second is the associative temporal memory base upon the tagged packet of dynamic data-driven scheme. Associative temporal memory was used to keep the temporal data to decrease the times of accessing the external memory. The third is the various compound instructions. The compound instructions were taken full advantage of to improve the throughput performance through the decrease of the number of operators. Since the throughput performance of history sensitive process is dependent upon the response time, the pipelined and current parallelism were exploited and the compound instructions were employed to reduce the length of the critical path. In addition, to reduce the memory requirement, the scattered block data were compacted and then they were stored into the associative temporal memory. Also, a one-bit plane was separated into top 8-bit and low 8-bit, then a lookup table only composed of 256 elements was demanded.

Concerning intra-frame coding, the scalable implementation of the proposed ABTC scheme was conducted. The results indicated that over 30 VGA frame images per second were achieved using 4 processors. The speedup ratio up to 1.6 was obtained by 8 processors. Although it is not increased proportionally along with the increment in number of processors, it was the best performance for the practical implementation due to the limitation of data dependencies inherent in the proposed scheme.

With respect to inter-frame coding, two types of input data structures had been explored. The experimental results indicates that the block-order input enables better throughput performance than the original scanline-order input. Moreover, the response time of block-order input is not changeable along with the larger size of image while that of scanline-order input becomes increased. Nevertheless, block-order input requires an additional module to transform the original scanline-order input into a sequence of 4x4 block images. To this end, a compound reference instruction of the line buffer was devised to access previous three lines of data simultaneously by virtue of the last line of data over each 4x4 block image. Experimental evaluation shows that around 60 VGA frames per second can be obtained on average utilizing block-order input which is twofold as many as that of previous frame [22]. The proposed scheme implemented on the data-driven chip-multiprocessor can therefore empower real-time coding for semi-motion picture applications.

5.1.3 Extension of This Work

In this subsection, the extension of this work towards algorithm and implementation will be discussed in detail.

(I) Algorithm Level

The performance of the proposed scheme can be improved by combining the coding approaches for bit plane. This is due to the fact that the 16-bit map of each block contains the redundancies which leaves the space to further increase the compression ratio.

Moreover, the proposed ABTC scheme employs the mean error coding approach to encode the pattern block. While the differences of absolute moment over each block image could be utilized instead of current approach so as to further increase the bit rate performance. The difference of absolute moment refers to the difference between the absolute value of a mean error and the absolute moment (AM). The number of bits for the difference of the absolute moment is thus less than that of mean error.

The inter-frame prediction approach used in the proposed ABTC scheme is very simple as the current block is only compared to its counterpart in the previous frame. While the motion prediction efficiency is degraded for the motion-intensive video sequences like Foreman or Salesman. To increase the motion prediction performance for motion-intensive video sequences, the binary block partitioning method with variable-size block matching [72] could be appended to the proposed scheme at the cost of more expensive computation.

Accordingly, there are large spaces and possibilities to continuously perfect the proposed ABTC scheme and extend its multimedia application domains based upon achieved research results so far.

(II) Implementation Level

In this study, a compound reference instruction of the line buffer has been proposed to transform the original scanline-order input data into a block-order one, while its circuit-level architecture was remained.

For future work, a logic-gate implementation of the proposed compound reference instruction of the line-buffer will be first completed and then its cost-performance will be evaluated. After that, a practical video coding system on a USB-DDMP chip will be established for real-time applications based on the proposed scheme and its data-driven parallel implementation.

5.2 A Framework for Future Directions

In this section, future directions for this study which will be hoped to motivate the researchers to undertake this field are outlined as follows.

5.2.1 How to Objectively Evaluate Image Quality

In this research, the image quality is evaluated in terms of the mean squared error (MSE) between the reconstructed luminance values and its original ones by virtue of the existing peak-signal-to-noise (PSNR) metric. Good reconstructed images typically have PSNR values of more than 30 dB.

However, PSNR does not always represent the reconstructed image quality perceived by the observer. This is because it does not consider the feature of human vision system (HVS). Accordingly, a superior metric which includes the HVS factor is desired to evaluate the perceived fidelity of the reconstructed image.

5.2.2 How to Efficiently Remove Interframe Redundancies

Motion prediction is essential to improve the compression efficiency. In the proposed ABTC scheme, the BTC-based motion prediction approach is actually a simple block-based difference coding. Since a current 4x4 block image is only compared to the one with the identical position in its previous frame, it is very simple in contrast to the motion estimation and compensation (ME/MC) widely used in the existing video coding standards such as H.263/H.264 and MPEG2/MPEG4. The effectiveness of this BTC-based motion prediction method has been proved by the semi-motion pictures like Missa video sequence. While it is not as good as ME/MC for motion-intensive video sequence like Foreman or Salesman due to its simplicity.

Nevertheless, both motion prediction approaches mentioned above are not optimum. This is due to the fact that the motion objects in the video sequence are not severely assortative with the block shape. So that the temporal redundancies can not be eliminated completely. A more efficient motion prediction approach is expected to meet very low-bit rate applications such as mobile communications.

5.2.3 How to Successfully Transmit Video Via WSN

With the emergence of wireless sensor network (WSN), how to successfully transmit the video images captured by multiple sensors via WSN is a future research direction. Since each sensor over WSN suffers from the low computing power and limited communication bandwidth, video coding algorithms targeted at WSN should possess low computational complexity and high compression efficiency. Moreover, the sensors are battery-powered so that low-power mobile platform is required. This is because frequently changing the battery is not feasible.

The proposed ABTC scheme is very simple because of no frequency transform.

Also, the data-driven multimedia processor is a low-power software implementation platform owing to its unique architecture, namely, dynamic data-driven scheme and self-timed pipelined circuits. So that research results presented in this thesis are promising for the WSN video communication application.

While WSN features changeable bandwidth and varied delay time, hence a video coding scheme should possess resilience to occasional packet loss and ability to quality recover from catastrophic failures. Accordingly, the robust error resilience and quality recover tools are required to be integrated into the proposed scheme.

5.2.4 How to Effectively Realize 3-D Image/Video Coding

Nowadays, users are expecting the three-dimensional (3-D) video streaming in place of existing two-dimensional (2-D) video steaming for the real-time communication applications such as person-to-person teleconferencing and remote surveillance. This is due to the fact that 3-D video can provide us with deep information of objects. Deep information can make our observation more vivid and distinct. Currently, deep information has been utilized in robotic walking and ship navigation as well as product inspection of industry. While ubiquitous consumer appliances like cars and mobile portable devices like laptop and mobile phone have not equipped with 3-D sensor until now, we have not enjoyed the vivid 3-D world supplied by the popular devices.

3-D image/video sequence includes twofold entropy information as many as monocular image/video sequence. Provided that existing coding standards are used to encode/decode two images/video sequences independently, the storage capacity and communication bandwidth will be double. In order to reduce the bandwidth requirement and storage capacity, intensive researches have been concentrated on the disparity-estimation and compensation method (DE/DC). This method takes full advantage of unique features, namely, binocular redundancies inherent in the stereo pairs captured by two sensors with slightly different directions. So that most of efficient stereo image/video coding schemes are based upon the DE/DC approach so far.

In order to cater to 3-D video communication and entertainment market demand, the proposed scheme will be extended to 3-D image and video coding by combining a disparity estimation/compensation (DE/DC) approach. Nevertheless, there are two major issues to be addressed for DE/DC-based stereo video coding. One is how to decide upon which one is more effective to use to decrease the redundancies inherent in the target images between DE and ME. Another one is that how to encode the disparity map can obtain the desired bit rate. This is because the characteristic of disparity map is different from the natural image, so that the generic coding approaches are not suitable for disparity map. While encoding the disparity map is essential to achieve low-bit rate performance, more efficient coding methods adaptive to its unique features of disparity map are expected in the future study.

The advanced works outlined above will contribute to establishment of 3-D flexible video communication systems with the high compression gains and error resilience tools for video transmission applications. Moreover, the images reconstructed by these video systems can be evaluated by the new metrics which include the characteristics of human vision system. As a consequence, such 3-D video surveillance systems over WSN are promising to be appeared in the airports, parking lots, markets and households in the future.

References

[1] G. J. Sullivan, “Video compression-from concepts to the H.264/AVC standard,” IEEE Proc. vol. 93, no. 1, pp. 18-31, Jan. 2005.

[2] I. Richardson, “H.264 and MPEG-4 video compresson: Video coding for next-generation multimedia,” 2003 John Wiley & Sons.

[3] Toshiba Corp.,“Introduction to the MeP (media embedded processor),” http://www.semicon.toshiba.co.jp/eng/prd/micro/doc/pdf/bce0043a.pdf, Jan. 2005.

[4] S. Okada, Y. Matsuda, T. Watanabe, and K. Kondo, “A single chip Motion JPEG codec LSI,” IEEE Trans. Consum. Elec., vol. 43, no. 3, pp. 418-422, Aug. 1997.

[5] H. Yamauchi, S. Okada, K. Taketa, and T. Ohyama, “A single chip JPEG2000 encode procesor capable of compressing D1-images at 30 frame/sec without tile division,” IEICE Trans. Elec., vol. E87-C, no. 4, pp. 448-456, Apr. 2004.

[6] N. Ahmed, T. Natrajan, and K. R. Rao, “Discrete consine transform,” IEEE Trans. Comput. vol. C-23, no. 1, pp. 90-93, Dec. 1984.

[7] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, “Video coding with H.264/AVC: tools, performance, and complexity,” IEEE on circuits and systems magazine, first quarter, pp. 7-26, 2004.

[8] T. Sikora, “Trends and Perspectives in image and video coding,” IEEE Proc. vol. 93, no. 1, pp. 6-17, Jan. 2005.

[9] E. J. Delp and O. R. Mitchell, “Image compression using block truncation coding,” IEEE Trans. Commun., vol. 27, no. 9, pp. 1335-1342, Sep. 1979.

[10] R. M. Gray, “Vector quantization,” IEEE ASSP Magazine, vol. 1, no. 2, pp. 4-29, Apr. 1984.

[11] M. Polvere and M. Nappi, "Speed-up in fractal image coding: comparison of methods," IEEE Trans. Image Processing, vol. 9, no. 6, pp. 1002-1009, Jun. 2000.

[12] C. W. Chao, C. H. Hsieh, and P. C. Lu, "Image compression using modified block truncation coding algorithm," Sig. Proces.: Image Commun 12. pp. 1-11, 1998.

[13] A. Dasu and S. Panchanathan, "A survey of media processing approaches," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 8, pp. 633-645, Aug. 2002.

[14] J. Hanson, "H.264 video encoding with Stretch's S5000 software configurable processor," http://www.videsignline.com/howto/showArticle.jhtml, Oct. 2005.

[15] G. Hinton et al., "The microarchitecture of the Pentium 4 processor," Intel Corp. Intel Technol. J., 1st Quarter 2001.

[16] Nagendra, "OMAP5910 BSP for VxWorks," http://www.mistralsoftware.com/html /services/downloads/whitepapers/OMAP5910_BSP_for_VxWorks.pdf, Oct. 2005.

[17] "Blackfin car telematics platform brings low cost telematics to the mass market," http://www.analog.com/UploadedFiles/White_Papers/726290983CTP_WP.pdf, Oct. 2005.

[18] A. Vetro, H. Sun, and Y. Wang, "MPEG-4 rate control for multiple video objects," IEEE Trans. Circuits Syst. Video Technol., vol. 9, no. 1, Feb. 1999.

[19] Y. Chang and M. Salim BEG, "Video over TETRA employing MPEG4 visual coding standard," IEICE Trans. Commun., vol. E87-B, no. 4, pp. 990-998, Apr. 2004.

[20] D. T. Lee, "JPEG2000: Retrospective and new development," IEEE Proc. vol. 93, no. 1, pp. 32-41, Jan. 2005.

[21] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthr, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits and Systems, vol. 13, no. 7, pp. 560-576, Jul. 2003.

[22] X. Yu and M. Iwata, "A Fast Video Coding Scheme and Its Data-Driven Parallel

Implementation," International Conference on Next Era Information, Networking NEINE'04, Kochi, pp. 406-413, Sep. 28, 2004.

[23] V. Udpikar and J. Raina, "BTC image coding using vector quantization," IEEE Trans. on Commu., vol. 35, no. 10, pp. 352-356, Oct. 1987.

[24] G. Arce and N. Gallagher, Jr., "BTC image coding using median filter roots," IEEE Trans. on Commu., vol. 31, no. 6, pp. 784-793, Jun. 1983.

[25] Y. Wu and D. Coll, "BTC-VQ-DCT hybrid coding of digital images," IEEE Trans. on Commu., vol. 39, no. 9, pp. 1283-1287, Sep. 1991.

[26] P. Franti and O. Nevalainen, "Block truncation coding with entropy coding," IEEE Trans. on Commu. vol. 43, no. 234, pp. 1677-1685, Feb.- Apr. 1995.

[27] M. Kamel, C. T. Sun, and L. Guan, "Image compression by variable block truncation coding with optimal threshold," IEEE Trans. Sig. Proces., vol. 39, no. 1, pp. 208-212, Jan. 1991.

[28] P. Nasiopoulos, R. K. Ward, and D. J. Morse, "Adaptive compression coding," IEEE Trans. on Commu. vol. 39, no. 8, pp. 1245-1254, 1991.

[29] S. Olsen, "Block truncation and planar image coding," Pattern Recognition Letters 21, pp. 1141-1148, 2000.

[30] N. Efrati, H. Liciztin, and H. C. Mitchell, "Classified block trancation coding-vector quantization: An edge sensitive image compression algorithm," Signal Processing: Image Commun. vol. 3, no. (2-3), pp. 275-283, 1991.

[31] Q. Kanafani, A. Beghdadi, and C. Fookes, "Segmentation-based image compression using BTC-VQ technique," IEEE Proc. of ISSPA 2003, vol. 1, no. 7, pp. 113-116, Jul. 2003.

[32] K. A. Wen and C. Y. Lu, "Hybrid vector quantization," Opt.Eng. vol. 32, no. 7, pp. 1496-1502, 1993.

[33] S. Horbelt and J. Crowcroft, "A hybrid BTC/ADCT video codec simulation bench,"

Seventh International Workshop on Packet Video, Brisbane, Australia, Mar. 1996.

[34] D. J. Healy and O. R. Mitchell, “Digital video bandwidth compression using block truncation coding,” IEEE Trans. on Commu., vol. COM-29, no. 12, pp. 1809-1817, Dec. 1981.

[35] M. D. Lema and O. R. Mitchell, “Absolute moment block truncation coding and its applications to color images,” IEEE Trans. Commun., vol. 32, no. 10, pp. 1148-1157, Oct. 1984.

[36] K. Ma and S. Rajala, “New Properties of AMBTC,” IEEE signal Processing letters, vol. 2, no. 2, Feb. 1995.

[37] A. K. Al-Asmari, S. Dave, and S. C. Kwatra, “Low complexity video compression algorithm using AMBTC,” Proc. of IEEE Military Communication Conference, Atlantic City, NJ, pp. 1241-1245, 31 Oct. - 3 Nov. 1999.

[38] H. Takata, T. Watanabe, T. Nakajima, T. Takagaki, H. Sato, A. Mohri, A. Yamada, T. Kanamoto, Y. Matsuda, S. Iwade, and Y. Horiba, “The D30V/MPEG multimedia processor,” IEEE Micro, vol. 19, pp. 38-47, Jul./Aug. 1999.

[39] J. Huck, D. Morris, J. Ross, A. Knies, H. Mulder, and R. Zahir, “Introducing the IA-64 architecture,” IEEE Micro, vol. 20, pp. 12-23, Sep./Oct. 2000.

[40] M. Tremblay, J. Chan, S.Chaudhry, A. W. Conigliam, and S. S. Tse, “The MAJC architecture: A synthesis of parallelism and scalability,” IEEE Micro, vol. 20, pp. 12-25, Nov./Dec. 2000.

[41] C. Basoglu, W. Lee, and J. S. O’Donnell, “The MAP1000A VLIM mediaprocessor,” IEEE Micro, vol. 20, pp. 48-59, Mar.- Apr. 2000.

[42] M. Berekovic, P. Pirsch, T. Selinger, K.-I. Wels, C. Miro, A. Lafage, C. Heer, and G. Ghigo, “Architecture of an image rendering co-processor for MPEG-4 systems,” in Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors, pp. 15-24, 2000.

[43] Y. Kang, J. Torrellas, and T. S. Huang, "Use IRAM for rasterization," in Proc. IEEE Int. Conf. Image Processing, vol. 3, pp. 1010-1013, 1998.

[44] G. Lauterbach et al., "UltraSPARC-III: A 3rd generation 64 b SPARC microprocessor," in Proc. IEEE Int. Solid-State Circuits Conf., pp. 410-411, 2000.

[45] T. Horel and G. Lauterbach, "UltraSPARC-III: Designing third-generation 64-bit performance," IEEE Micro, vol. 19, pp. 73-85, May/Jun. 1999.

[46] M. Tremblay and J. M. O'Connor, "UltraSparc I: A four-issue processor supporting multimedia," IEEE Micro, vol. 16, pp. 42-50, Apr. 1996.

[47] M. Tremblay, J. M. O'Connor, V. Narayanan, and L. He, "VIS speeds new media processing," IEEE Micro, vol. 16, pp. 10-20, Aug. 1996.

[48] D. A. Draper, et al., "An X86 microprocessor with multimedia extensions," in Proc. IEEE Int. Solid-State Circuits Conf., pp. 172-173, 1997.

[49] R. B. Lee, "Subword parallelism with MAX-2," IEEE Micro, vol. 16, pp. 51-59, Aug. 1996.

[50] S. Wong, S. Cotofana, and S. Vassiliadis, "Multimedia enhanced general-purpose processors," in Proc. IEEE Int. Conf. Multimedia Expo, vol. 3, New York, pp. 1493-1496, 2000.

[51] R. Lee and J. Huck, "64-bit an multimedia extensions in the PA-RISC 2.0 architecture," in Proc. Compcon'96, pp. 152-160, 1996.

[52] T. Lanier, "NEON and OptimoDe technologies: The ARM approach to signal processing," vol. 4, no. 2, pp. 22-23, 2005.

[53] P. Biswas et al., "SH-5: The 64-bit SuperH architecture," IEEE Micro, vol. 20, no. 4, pp. 28-39, Jul. 2000.

[54] F. Arakawa et al., "SH4 RISC Multimedia Microprocessor," IEEE Micro, vol. 18, no. 2, pp. 26-34, Mar.- Apr. 1998.

[55] A. Peleg and U. Weiser, "MMX technology extension to the Intel architecture,"

IEEE Micro, vol. 16, no. 4, pp. 42-50, 1996.

[56] "OMAP5912 applications processor," http://focus.ti.com/lit/ds/sprs231d/sprs231d.pdf, Mar. 2005.

[57] D. Katz and R. Rivin, "Embedded media processing requires new strategies," http://wwww.analog.com, Nov. 2003.

[58] H. Terada, S. Miyata and M. Iwata, "DDMP's: self-timed super-pipelined data-driven multimedia processors," Proc. IEEE, vol. 87, no. 2, pp. 282-296, Feb. 1999.

[59] G. Côt'e, B. Erol, M. Gallant, and F. Kossentini, "H.263+: video coding at low bit rates," IEEE Trans. Circuits and Systems, vol. 8, no. 7, pp. 849-866, Nov. 1998.

[60] X. Yu and M. Iwata, "An adaptive block truncation coding and its data-driven parallel implementation," submitted to IEICE Trans. INF. & SYST.

[61] X. Yu and M. Iwata, "A video coding algorithm based on classified block truncation coding and its data-driven parallel implementation," International Conference on Next Era Information Networking NEINE'05, ShangHai, China, pp. 153-164, Sep. 4-5, 2005.

[62] "Intel Pentium D processor," http://www.intel.com/products/processor/pentium_d/prodbrief800.pdf, Jan. 2006.

[63] S. Tsuneishi, X. Yu, and M. Iwata, "Data-driven parallel implementation of a BTC-based image compression," International Conference on Next Era Information Networking NEINE'05, ShangHai, China, pp. 534-541, Sep. 2005.

[64] Y. L. Chan and W. C. Siu, "New adaptive pixel decimation for block motion vector estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 2, pp. 113-118, Feb. 1996.

[65] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 6, pp. 313-317, Jun. 1996.

[66] C. H. Lin and J. L. Wu, "A lightweight genetic block-matching algorithm for video coding," IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 2, pp. 386-392, Feb. 1998.

[67] H.-S. Wang and R. M. Mersereau, "Fast algorithms for the estimation of motion vectors," IEEE Trans. Image Processing, vol. 8, no. 3, pp. 435-438, Mar. 1999.

[68] K. W. Wong, K. M. Lam, and W. C. Siu, "An efficient low bit-rate video-coding algorithm focusing on motion region," IEEE Trans. Circuits Syst. Video Technol., vol. 11, no. 10, pp. 1128-1134, Oct. 2001.

[69] X. Yu and M. Iwata, "Image compression based on BTC-DPCM and its data-driven parallel implementation," IEEE International Conference on Image Processing (ICIP05), Genova, pp. 813-816, Sep. 11-14, 2005.

[70] jj2000-4.1, http://jj2000.epfl.ch//jj_download, Oct. 2002.

[71] H.264/AVC codec, http://iphome.hhi.de/suehring/tml, Apr. 2005.

[72] M. Servais, T. Vlachos, T. Davies, "Motion-compensation using variable-size block-matching with binary partition trees," IEEE International Conference on Image Processing (ICIP05), Genova, Sep. 11-14, 2005.