

H.264 Video Encoding Algorithm on Cell Broadband Engine

Ashish Jagmohan, Brent Paulovicks, Vadim Sheinin, Hangu Yeo
 Department of Multimedia Technologies, IBM TJ Watson Research

Mailing address: IBM TJ Watson Research Centre, P. O. Box 218, Yorktown Heights, NY-10598, USA
 {ashishja, ovicks, vadims, hangu}@us.ibm.com

Abstract—In this paper we present an implementation of an H.264 video encoding algorithm on a Cell Broadband Engine (CBE), for the application of high-quality video surveillance. The proposed system aims to encode three channels of a standard-definition (720×480) video stream at 30 frames per second with a target bit-rate of 2 Mbps. The presented encoder is compliant with the main-profile of the H.264 standard, and uses a learning-theoretic mode selection algorithm as an alternative to brute-force rate-distortion optimized mode selection, enabling significantly reduced computational complexity. The CBE offers an aggregate of 204.8 GFlops of computing power, at 3.2 GHz, in 8 Synergistic Processor Elements (SPEs), each with 128-bit wide vector processing capability. The SPEs are under the control of a central Power Processor Element (PPE) which has its own 128-bit vector processing unit and all units are connected by an on-chip broadband bus with 25.6 GB/s bandwidth capacity and an I/O bus providing 50 GB/s. This combination of processing units and high-speed internal buses is ideally suited for the target application of multi-channel real-time H.264 video encoding. The proposed system employed only the standard tools provided with the CBE toolkit, without resorting to customized assembly level programming.

I. INTRODUCTION

There is an obvious need today for large-scale enterprise class Digital Video Surveillance (DVS) solutions, which can assist governments, local authorities and other organizations to improve safety and security. Such systems are currently finding applications in industries and organizations as diverse as transportation, retail, finance and government. Another emerging trend is the rapid growth in the number of cameras constituting such systems. A good example of this is the Safety City project in Moscow wherein the city government plans to finish installation of 90000 cameras around the city by 2007, of which around 6000 have been already installed.

The chief concerns that need to be addressed while designing a large-scale DVS system are as follows. The solution should be scalable, such that it can be expanded to address a growth in the number of surveillance devices on an on-demand basis, and can be used to address installations with a large number of such devices. The second key concern is that of low total cost of ownership (TCO). This requires that the hardware and software complexity and maintenance costs be feasibly low. An important aspect of this is that the system allow efficient compression of the captured video such that the storage costs can be kept low. Other key concerns include reliability, i.e. ensuring that the system be redundant and fault-

tolerant, and programmability, to allow the use of sophisticated video encoding algorithms and image/video analytics software.

Current video surveillance systems are typically based on the use of digital video recorders (DVR), which are essentially PCs with video encoding boards. These typically employ MPEG2/MPEG4 [1] compliant video encoding algorithms. These solutions have several shortcomings vis a vis the preceding requirements. Since each DVR can typically process only a few channels of video, hundreds of servers would be required for installations with thousand of cameras. This dramatically increases the TCO for such solutions. Secondly the use of MPEG-based video encoding algorithms results in a significant loss in compression efficiency when compared to the state-of-the-art H.264 [2] standard. Finally, it is difficult to build reliable and flexible DVS solutions from PC-based DVRs.

In this paper, we present an implementation of an H.264 video encoding algorithm on a Cell Broadband Engine (CBE), for the purpose of digital video surveillance. In particular, we will focus on the design of a single CBE based H.264 video encoding system which aims to encode three channels of a standard-definition (SD) video stream at 30 frames per second at a target bit-rate of 2 Mbps. The proposed architecture can be used to serve as the core of a large-scale DVS system which would be scalable, have far smaller TCO than DVR based solutions, and allow high programmability.

Cell Broadband Engine has enormous power for processing broadband content for media, entertainment and video game industries. In 2001 Sony, IBM and Toshiba started to co-develop Sony's next generation PlayStation¹ CPU at a joint design center in Austin, Texas. Sony had a vision of bringing virtual reality to the experience of their game players. This defined the architecture of the CBE - highly parallel, very fast, with huge memory throughput. The three companies disclosed the details on the 64-bit Power ArchitectureTM based CBE in the International Solid-State Circuits Conference (ISSCC) in 2005 [3]. This multi-core chip contains one 64-bit Power Architecture based core and eight 32-bit Synergistic Processor Elements (SPEs). The chip is 235 mm^2 device fabricated with 90nm silicon-on-insulator (SOI) technology with 250M total number of transistors integrated.

The H.264 standard provides state-of-the-art video compression with gains of up to 50% in compression efficiency, when compared to previous standards such as MPEG-4. The key

¹PlayStation is a trademark of Sony Computer Entertainment Inc.

features incorporated in the H.264 standard, from which it derives most of its compression gain, are the use of efficient arithmetic coding, the use of quarter-pixel accurate motion vectors for motion compensation, and the use of several different macroblock prediction modes and block sizes for encoding 16×16 macroblocks. A key concern, in this context, is the large increase in encoder computational complexity (as compared to the MPEG standards) resulting from the need for macroblock mode selection. Large computational complexity is undesirable as it necessitates a corresponding increase in hardware resources required for achieving real-time compression. In this paper, we will describe the design of a computationally efficient H.264 encoder which utilizes learning-theoretic mode-selection algorithms to significantly reduce encoder computational complexity, without sacrificing compression efficiency.

The organization of the paper is as follows. Section 2 briefly describes the advantages of the proposed CBE based video compression system. Section 3 briefly describes the learning-theoretic mode selection algorithms for efficient H.264 compression. Section 4 describes the Cell Broadband Engine Architecture (CBEA) in greater detail. Section 5 provides a brief description of the issues involved in porting the H.264 encoder implementation onto the CBE. Finally, Section 6 concludes the paper with a brief presentation of results.

II. PROPOSED CELL BROADBAND ENGINE BASED SYSTEM

We propose to design a CBE based H.264 video encoder which can be used as the core constituent of a DVS system. The central motivation for this is that such a core provides the possibility of constructing a large-scale DVS solution that has excellent scalability, low TCO, and is reliable and programmable. In this section, we will briefly discuss these advantages.

Scalability is achieved by the possibility of using the CBE with IBM blade centers. Given that each CBE can encode three channels of SD video at 30 frames per second (fps), a full capacity blade center containing seven blades each carrying two CBEs can encode 42 streams of SD video at 30 fps. This is very high—compare, for example, to a typical PC carrying six PCI boards each of which can encode a single SD video channel. Thus the number of servers required by a CBE based solution would be almost an order of magnitude lower than that required by a PC based DVS system.

Total cost of ownership would be reduced in two ways by such a system. Firstly, the large reduction in the number of servers, described above, would directly reduce the hardware maintenance and system administration costs. This is accentuated by the use of the Linux operating system with the CBE, which reduces TCO substantially compared to Windows machines. Secondly, the use of the H.264 encoding algorithm which is upto two times more efficient than previous standards reduces storage costs by upto half. The importance of this can be seen readily. A single camera running at 4 fps and 0.4 Mbps would generate 4.3 GB of data in a day. Thus a large-scale DVS system with 3000 cameras would generate 13 terabytes of data a day, which may be required to be stored for a period

of months or even years. Even at the cost of a few cents per megabyte of storage, it is clear that storage costs would play a significant role in the TCO.

Finally, the CBE is fully programmable, and the blade center has built in redundancy, in the form of redundant ethernet switches for example. Thus, a CBE based H.264 video encoding system seems like an ideal choice for building large-scale DVS solutions. In the remainder of this paper we will discuss the design of a low-complexity H.264 encoder, and describe how it was implemented on the CBE.

III. TIME-EFFICIENT H.264 IMPLEMENTATION

Our implementation of the H.264 encoder contains all main-profile tools with the exception of interlaced coding and weighted prediction. In addition it employs a subset of all possible block coding modes, which will be further described in this section. We now provide a brief description of the time-efficient mode selection algorithm employed in our H.264 implementation. Details can be found in [4].

The main-profile of the H.264 standard defines two main classes of prediction modes for 16×16 luma macroblocks, namely the intra prediction modes and the inter prediction modes. Chroma block mode selection is implicitly decided by the selected luma block mode. There are two intra prediction modes—the Intra16 (I16) mode and the Intra4 (I4) mode. The I16 prediction mode generates a causal 16×16 predictor for the current macroblock using neighboring macroblocks from the current frame. The I4 prediction mode generates causal 4×4 predictors for each of the 16 constituent 4×4 blocks of the current macroblock. Several intra prediction submodes are provided within these two mode classes, including nine I4 prediction modes and four I16 prediction modes.

The standard also provides two main inter prediction modes, one of which is invoked depending on the frame type. P frames utilize single reference-list prediction, while B frames utilize two-list prediction. The standard allows the 16×16 macroblock to be partitioned into smaller blocks of varying sizes for the purpose of motion compensation. In the presented implementation we restrict ourselves to the use of 16×16 block-based motion compensation to reduce time complexity. Given this restriction, there is only one available P prediction mode (denoted the P16 mode) and four available B modes, namely list0 B prediction (B_L0), list 1 B prediction (B_L1), bidirectional B prediction (B_Bi), and direct mode prediction (B_Dir).

Macroblock mode selection involves selecting the prediction mode class, and further selecting the specific prediction mode from that class, which results in the best compression efficiency for coding the current macroblock. Thus mode selection involves, (1) Choosing between inter and intra prediction, (2) If intra prediction is to be used, choosing between I4 and I16 modes, (3) If P prediction is used, finding the best motion compensated macroblock prediction, and (4) If B prediction is to be used, choosing the most efficient of the four B modes, and finding the best motion-compensated macroblock predictor.

The main shortcoming of the conventional H.264 mode selection algorithm, as implemented by the reference encoder,

is that all possible predictor modes are evaluated during mode selection, for comparison using a Lagrangian functional [5]. Each such mode evaluation is quite costly in terms of computational complexity, requiring the current macroblock to be differentially encoded with respect to the generated predictor, with the error being transform coded, quantized and CABAC coded. In our implementation, we make use of a learning-theoretic approach to select from among the main prediction classes, and to further select the best prediction mode within the selected class. The learning-theoretic algorithm is trained offline using a set of standard training sequences, and only a small amount of computation is required, to compute the learned discrimination functions, during encoding. Thus the time complexity of mode selection, which is a significant constituent of the overall time complexity of the reference H.264 encoder, is significantly reduced.

Fig. 1 shows a simplified block diagram of the proposed mode selection algorithm. First, a fast motion compensation algorithm, termed the stabilized diamond search (SDS) algorithm (akin to the diamond search algorithm proposed in [6]), is used to find the best motion compensated P/B mode predictors. For the case of B frames, the best 16×16 B mode is selected on the basis of the motion-compensation cost associated with each, which consists of a weighted combination of the motion-vector encoding cost and the sum of absolute differences (SAD) of the current macroblock and the predictor macroblock. Subsequently, a learning-theoretic classification algorithm, which operates on a set of transform domain features extracted from the macroblock and the P/B mode predictor, is used to select between the inter and intra prediction classes. Finally, if intra prediction is selected, a second classification algorithm is used to select between the I4 and the I16 prediction modes, and the best submode is selected by a time-efficient search algorithm which utilizes partial-SAD values.

Both the inter-intra and I4-I16 decisions are made using supervised binary classification [7] using classifier trees [8]. As detailed in [4], classification is performed as follows. Prior to classification the 16×16 current macroblock m is downsampled to a 4×4 array, denoted m_4 . Denoting the 4×4 Hadamard transform of m_4 as $\mathbf{T}_H m_4$, the following set of features are used for I4-I16 discrimination: (1) The macroblock high frequency content $f_1 = \sum_{i=2}^4 \sum_{j=2}^4 |\mathbf{T}_H m_4(i, j)|$, (2) The macroblock horizontal frequency content $f_2 = \sum_{j=2}^4 |\mathbf{T}_H m_4(0, j)|$, and (3) The macroblock vertical frequency content $f_3 = \sum_{i=2}^4 |\mathbf{T}_H m_4(i, 0)|$. In addition, for intra-inter classification, the following features were used: (3) The absolute sum of the transform coefficients of the prediction error between the downsampled current macroblock m_4 , and the downsampled motion compensated predictor p_4 , i.e. $f_4 = \sum_{i=1}^4 \sum_{j=1}^4 |\mathbf{T}_H(m_4 - p_4)(i, j)|$.

IV. CELL BROADBAND ENGINE ARCHITECTURE

CBE is a very powerful processor which is highly suitable for game and multimedia processing. CBE is the result of a collaboration between Sony Computer Entertainment Inc. (SCEI), IBM Corporation (IBM) and Toshiba Corporation

(Toshiba) started in 2001, and the architecture of the processor has been presented in the International Solid-State Circuits Conference (ISSCC) in 2005 [3].

CBE is a heterogeneous chip multiprocessor with a 64-bit Power Processor Element (PPE) with L2 cache and eight specialized coprocessors (Synergistic Processor Elements (SPEs)) on a SIMD-based architecture with 32-bit wide instructions Fig. 2. The PPE has its own 128-bit vector processing unit and all units are connected by an on-chip broadband bus. Each SPE is a private system-on-chip with the processing unit connected to limited 256KB of Local Storage (LS) which may contain program, stack, local data structures and DMA buffers. The eight SPEs provide tremendous computing power with an aggregate of 204.8 GFLOPS at an operating frequency of 3.2 GHz. The SPE SIMD engine provides 128-entry 128-bit SIMD register file and up to 16-way SIMD to exploit data parallelism. The PPE is connected to the SPEs through the internal Element Interface Bus (EIB). The EIB also connects to the L2 cache and Memory Interface Controller (MIC), and it can handle up to 96 bytes/cycle. The processing units (PPE and SPEs) access the system memory through the shared MIC which is connected to dual channels of Rambus XDR² providing total bandwidth in the range of 25GB/s. Memory access is performed via DMA-based interface (2×25.6 GB/s) initiated by either the PPE or an SPE. The DMA is managed by the software. Each DMA request can transfer in the range of 16 bytes and 16 KB of data. Communication with the rest of the system is done through the high speed FlexIO interface.

The SPE was designed with a compiled code focus from the beginning. The SIMD-optimized compiler for the SPE is based on IBM XL compiler technology optimized for the SPE architecture, and supports altivec-like intrinsics and limited auto-vectorization. A complete system simulator for the PowerPC (called Mambo) is also available to emulate the behavior of a CBE. The Mambo simulator let the user be able to get details of statistics in a system running a real workload, which may be unable to get from the real hardware.

V. H.264 IMPLEMENTATION ON CELL BROADBAND ENGINE

A straight forward mapping of multiple H.264 encoders onto a Cell processor might statically assign 'n' SPEs to 'm' encoders, constrained by the 8 available SPEs, Unfortunately, surveillance differs from the usual video application, say broadcasting or conferencing, in that the frame rates and sizes can vary from camera-to-camera and even from time-to-time from the same camera. It is not unusual for a low-interest camera to be sampled at 5 fps while a main thoroughfare is monitored at the full 30 fps. Likewise, the frame size may vary depending upon the scene under observation, SIF and SD being the two principle resolutions, and with time if an alarm occurs which switches a camera from SIF to SD. For surveillance applications, then, statically assigning SPEs to encoders would not make the most efficient use of the Cell's resources.

²XDR is a trademark of Rambus corp.

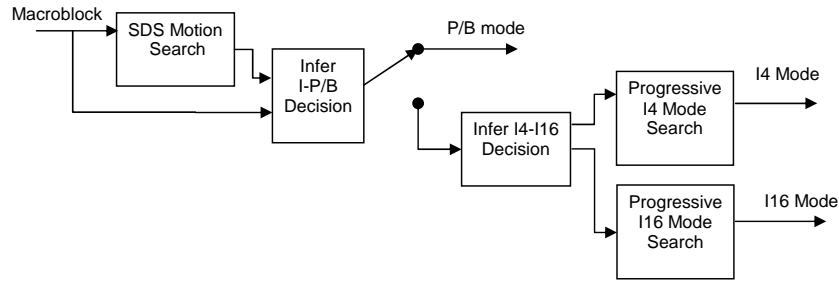


Fig. 1. Block diagram illustrating mode selection algorithm.

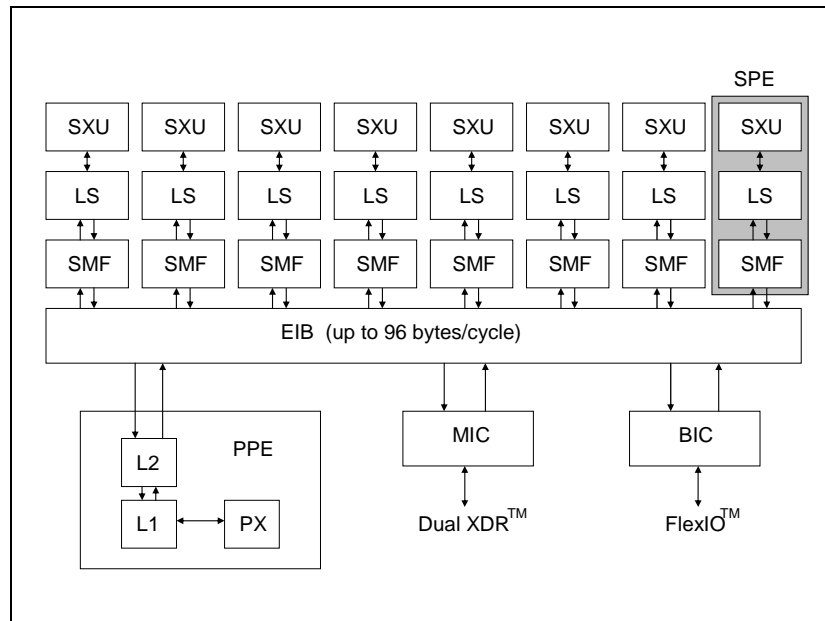


Fig. 2. CBE block diagram.

The approach we used was to assign an SPE to an encoder to complete a unit of work and then to be reassigned. The H.264 encoding process was divided into three basic functions: motion estimation (ME), transform and quantization (PI), and CABAC (AC). Each of these functions operates on an entire frame at a time and constitutes the unit of work for an SPE. The SPE is passed a command block which contains the operation to be performed, and pointers to the frame buffers on which it is to perform the function. As the SPE completes the function on a frame, it notifies the PPE which reassigns it to the next pending work unit. In this way, any mix of frame rates, frame sizes, and IPB stream structures can be managed automatically by the Cell.

As mentioned, the encoding process was divided into three functions. The first, motion estimation (ME) was, perhaps, the most difficult to port to an SPE due to the large amount of buffering this function requires. In our implementation, the range for motion estimation was limited to a window of up to eight macroblocks in the vertical direction but can search the full width of the frame in the horizontal direction. That

is, a window of at most eight rows of macroblocks from the reference frame is maintained inside the SPE and one row of macroblocks from the current frame is processed against it. As the ME function proceeds thru the frame, one row of macroblocks is fetched from both the reference and the current frames. Though several search heuristics are used to limit the number of SAD computations, search results are to quarter-pixel resolution. Besides motion estimation, this function also computes the heuristic used to select between intra and inter coding for a macroblock. This, along with the motion vectors and motion compensated prediction is returned to system memory after each row is processed.

As with the ME function, the second function, transform and quantization (PI), processes a frame by row of macroblocks. Each row of macroblocks from the current frame is read into the SPE along with the motion-compensated prediction for that row and the intra/inter decision made by the ME function. Though the intra/inter decision was made by ME, intra prediction needs further refinement to select either 16×16 or 4×4 mode which is, again, decided by a heuristic. After

this, an exhaustive search is made of either the four 16×16 possible predictions or the nine 4×4 predictions. From these predictions are generated the macroblock coefficients (transform and quantization) and the reconstructed frame (inverse quantization, inverse transform and loop-filter). Once PI has completed the reconstructed frame is available to be used as a reference for the next frame.

The last function is the CABAC encoding (AC) of the motion vectors and coefficients. Although not suited to a vector processor, it was necessary for an SPE to do this encoding as the PPE alone would not have the processing power to handle multiple streams along with all of the housekeeping chores it must do, e.g., orchestrating the flow of frames from multiple cameras through the SPEs for compression and handling the network traffic required to send the compressed results to a server for storage. Despite the serial nature of the CABAC code, the SPE is able to perform this function in approximately the same amount of time as the PPE.

VI. RESULTS

To analyze the performance of the presented H.264 encoder, three standard definition (720×480) 4:2:0 YUV sequences, namely, *cheers*, *mobile_and_calendar*, and *table_tennis* were encoded using three encoders—the proposed time-efficient encoder, the reference JM8.2 H.264 encoder, and a reference MPEG-4 encoder. 59 frames of each sequence were encoded using each of these encoders. A comparison of compression efficiency for the three codecs was done by comparing PSNR-Rate curves for each sequence.

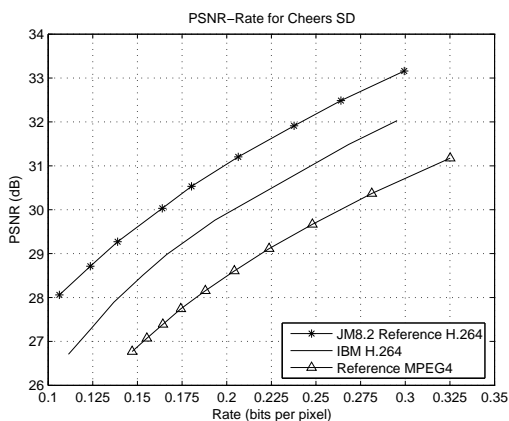


Fig. 3. PSNR-Rate plots for standard definition *cheers* sequence.

Figure 3 compares the convex hull of the PSNR-Rate performance curves for the three codecs for the *cheers* sequence. The rate is measured in terms of bits-per-pixel. For reference, 0.2 bits-per-pixel corresponds to a rate of approximately 3.1 Mbps at 30 fps. As can be seen, the IBM H.264 codec is about 1-1.5 dB inferior to the reference H.264 codec, and is 1.5-2 dB superior to the reference MPEG4 codec. Similar results for *table_tennis* and *mobile_and_calendar* are shown in Figures 4 and 5 respectively. We note that the loss in compression efficiency vis a vis the reference H.264 encoder is accompanied by a large gain in computational efficiency—a software implementation of the presented encoder, running

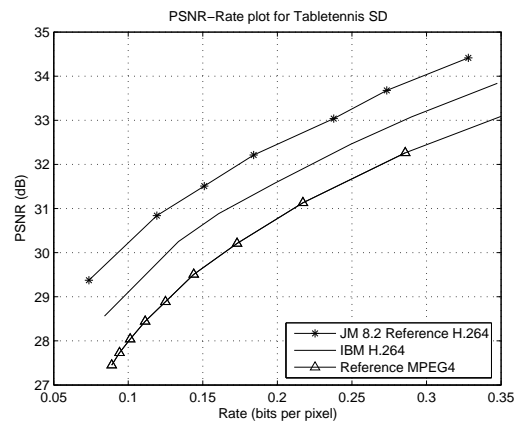


Fig. 4. PSNR-Rate plots for standard definition *table_tennis* sequence.

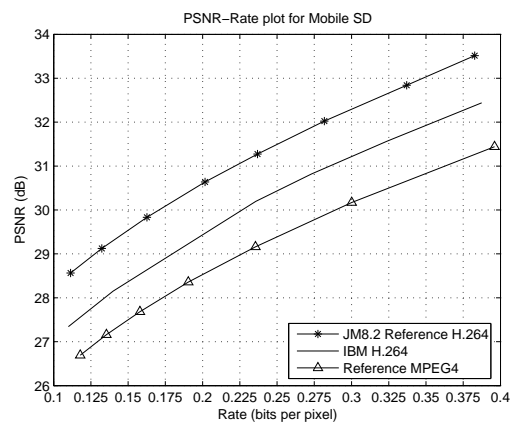


Fig. 5. PSNR-Rate plots for standard definition *mobile_and_calendar* sequence.

on a Windows based Pentium workstation, is about 40 times faster than a corresponding software implementation of the H.264 reference coder.

Table I shows the average time each function (ME,PI,AC) takes for the three different video sequences, each encoded at three different bitrates. These times were measured on a 2.4 GHz CBE. As expected, CABAC requires increasing amounts of time as the bit rate increases. Note that the processing required to produce the next reference frame (ME+PI) can be done well within the 33 millisecond frame period so that motion estimation for the next frame can begin as soon as the frame arrives. Virtually all of the encoding is performed by the SPEs with only a small amount of NAL unit processing being done by the PPE.

REFERENCES

- [1] ISO/IEC JTC 1/SC 29/WG 11, "Mpeg-4 video coding standard," 2000.
- [2] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.
- [3] D. Pham et al., "The design and implementation of a first-generation cell processor," 2005.
- [4] A. Jagmohan and K. Ratakonda, "Time-efficient learning theoretic algorithms for h.264 mode selection," in *Proc. IEEE Int. Conf. Image Processing*, 2004, pp. 749–752.

TABLE I
 PROCESSING TIMES ON 2.4 GHZ CELL BROADBAND ENGINE FOR CELL
 ENCODING FUNCTIONS.

	ME (ms)	PI (ms)	AC (ms)
table tennis (1 Mbps)	11.9	14.9	13.6
table tennis (2 Mbps)	11.4	15.6	17.4
table tennis (3 Mbps)	11.1	17.8	21.5
cheers (1 Mbps)	14.3	18.9	15.5
cheers (2 Mbps)	13.2	18.9	17.9
cheers (3 Mbps)	13.4	19.0	21.4
mobile (1 Mbps)	13.6	15.6	13.0
mobile (2 Mbps)	13.0	16.7	17.9
mobile (3 Mbps)	12.9	16.7	20.9

- [5] T. Wiegand, M. Lightstone, D. Mukherjee, T.G. Campbell, and S.K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging h.263 standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 182–190, Apr. 1996.
- [6] S. Zhu and K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, Feb. 2000.
- [7] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, Wiley, 2001.
- [8] P. Chou, "Optimal partitioning for classification and regression trees," *IEEE Trans. Pat. Anal. Mach. Intel.*, vol. 13, no. 4, pp. 340–354, 1991.