

PARALLEL RATE-DISTORTION OPTIMIZED INTRA MODE DECISION ON MULTI-CORE GRAPHICS PROCESSORS USING GREEDY-BASED ENCODING ORDERS

Ngai-Man Cheung, Oscar C. Au, Man-Cheung Kung, Xiaopeng Fan

Department of Electronic & Computer Engineering, Hong Kong University of Science & Technology

ABSTRACT

Rate-distortion (RD) optimized intra-prediction mode selection can lead to significant improvement in coding efficiency in intra-frame encoding. However, it would incur considerable increase in encoding complexity. In this paper, we investigate how multi-core Graphics Processing Units (GPUs) can be efficiently utilized to undertake the task of RD optimized intra mode selection in AVS and H.264 video encoding. Achieving efficient GPU-based intra mode decision, however, could be non-trivial. It is because the mode decision of the current block would depend on the *reconstructed data* of the neighboring blocks. Therefore, the coding modes of neighboring blocks would need to be computed first before that of the current block can be determined. This dependency poses challenge to computation on multi-core GPUs, which rely heavily on parallel data processing to achieve superior speedups. To address this issue, we analyze the data dependency in intra mode decision, and propose novel greedy-based encoding orders to achieve highly parallel processing. We also prove that the proposed greedy-based orders are optimal in terms of execution time. Experimental results suggest that the proposed GPU-based intra mode decision compares favorably to the counterpart implemented on a single-core CPU.

Index Terms— RD optimized intra-prediction, GPU, multi-core, greedy approach, parallel processing

1. INTRODUCTION

State-of-the-art video encoding algorithms employ rate-distortion (RD) optimized intra-prediction mode selection in intra-frame coding to achieve competitive coding performance [1, 2]. In RD optimized intra mode selection, the encoder may compute the RD costs of all the possible prediction modes, and select the one with the minimum RD cost to encode the current block. Each prediction mode corresponds to a different intra prediction direction. In general, RD cost calculation may involve computing the intra prediction residue, transformation and quantization on the prediction residue, inverse quantization and inverse transformation, and entropy coding of the quantized transform coefficients. These operations may need to be repeated for every candidate prediction mode when encoding each video block. Therefore, the computational complexity of RD optimized intra mode selection could be very significant [3].

Focusing on AVS/H.264 [1, 2] software-based encoders on personal computers (PC) or game consoles equipped with both Central Processing Units (CPUs) and programmable Graphics Processing Units (GPUs), this paper investigates how RD optimized intra mode decision can be efficiently implemented on GPUs. Modern GPUs

may consist of hundreds of highly decoupled processing cores offering immense computing power. For example, the NVIDIA GeForce 8800 GTS processor, which is used in our simulations, consists of 96 individual *stream processors* each running at 1.2 GHz [4]. The stream processors can be grouped together to perform Single Instruction Multiple Data (SIMD) operations suitable for data-intensive applications. With advances in programing tools, multi-core GPUs have emerged recently as co-processing units for CPUs to accelerate various signal processing applications (e.g., [5]).

Achieving efficient GPU-based intra mode decision, however, could be non-trivial. It is because intra mode decision tends to be sequential. Specifically, to evaluate various candidate prediction modes for the current block, it is necessary to have the *reconstructed data* of the neighboring blocks available to be used as reference pixels [1, 2]. Therefore, the coding modes of neighboring blocks would need to be first determined, and these blocks would be encoded and reconstructed accordingly. Then, prediction mode of the current block can be determined based on the reconstructed neighboring blocks. Such dependency between the coding modes of adjacent blocks would complicate the mapping of intra mode decision into multi-core architectures.

In this work, we analyze the dependency constraints in intra mode decision, and propose a strategy to determine the mode decisions of video blocks in parallel. In particular, we derive graphical models that capture the dependency relationship in intra mode decision. Based on the dependency models we propose to encode the blocks in novel orderings, so that highly parallel processing can be achieved. We also present a proof that the proposed encoding orders, which are based on the greedy approach, are optimal in terms of execution time. We remark that, as will be discussed, our work addresses the block dependency issue by proposing new encoding orders, rather than disabling certain candidate modes to remove some dependency. Therefore, our proposed strategy does not suffer any penalty on compression efficiency.

Some recent work has proposed to use GPU in video compression framework. [5] proposed a system where CPU and GPU work in parallel to accelerate video decoding. Others have proposed to use GPU to accelerate motion estimation, e.g., [6]. Some recent work has also proposed to accelerate intra prediction by processing the data in different orders within a *macroblock* [7–9]. On the contrary, the current work proposes to modify the block processing order within a *frame*, leading to high degree of parallel computation suitable for GPU implementation. Our previous work [10] has also proposed to facilitate intra frame encoding by re-arranging the block encoding order in an ad-hoc manner. On the contrary, the present work presents a thorough dependency analysis, and proposes to process video blocks following a greedy approach. While some results are similar to [10], by following a systematic design approach our results are amenable to analysis. In particular, we are able to show that our results are optimal. The differences between our proposed

This work has been supported in part by the Innovation and Technology Commission (project no. GHP/048/08) and the Research Grants Council (project no. RPC07/08.EG22) of the Hong Kong Special Administrative Region, China.

solution and *list scheduling* [11] (which has been used to address instruction scheduling in processors with multiple functional units) are discussed in the extended version of this paper [12]. Other work has proposed new image analysis algorithms for parallel implementation, e.g., [13].

The rest of this paper is organized as follows. In Section 2 we briefly review intra prediction. Section 3 presents a dependency analysis of intra decision, and proposes novel encoding orderings based on the greedy approach. Section 4 presents our simulation results. Finally, Section 5 concludes the work.

2. INTRA PREDICTION

Here we give a brief overview of AVS RD-optimized intra prediction, while that of H.264 can be found in [1]¹. In AVS, every 8×8 block can be intra predicted by one of the five prediction modes, each corresponding to a different prediction direction as depicted in Figure 1 [2]. To determine the optimal mode an encoder may compute the Lagrangian costs for all the coding modes, and select the one which achieves the minimum. The Lagrangian cost J can be computed as follows. In the “low complexity mode”:

$$J = SATD + \lambda(R_{header}), \quad (1)$$

where SATD is the sum of absolute transformed differences between the original current block and the prediction block, R_{header} is the estimated bits for the header information, and λ is the Lagrangian multiplier. In the “high complexity mode”, J is computed by

$$J = SSD + \lambda(R_{header} + R_{res}), \quad (2)$$

where SSD is the sum of square differences between the original and reconstructed current block, R_{header} and R_{res} are the encoded bits for header and quantized residual block. Note that neighboring *reconstructed* pixels are required to compute J in (1) and (2).

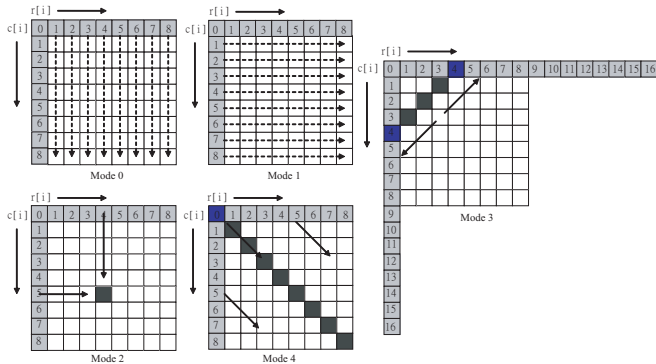


Fig. 1. Intra prediction in AVS 1.0 [2]: mode 0: vertical, mode 1: horizontal, mode 2: DC, mode 3: down-left, and mode 4: down-right. Here white squares denote the current block pixels and grey squares represent the neighboring *reconstructed* pixels.

3. PARALLEL RD OPTIMIZED INTRA MODE DECISION

3.1. Dependency Analysis

As discussed, since reconstructed pixels are used as reference in prediction, it is necessary that all the candidate reference blocks have been encoded and reconstructed first, before the RD cost of the current block can be computed. Therefore, by considering the prediction direction of each applicable mode, we can derive the dependency between the current block and its neighbors. For example, Figures 2(a) and 3(a) show the dependency relationships when

¹We focus on H.264 4×4 intra prediction in this paper, while the analysis can be readily applied to other prediction block sizes.

computing the intra prediction RD costs in AVS and H.264 respectively. A directed edge going from block A (*parent node*) to block B (*child node*) indicates that block B requires the reconstructed pixels from block A to determine the RD costs of various candidate modes. Therefore, block B cannot be encoded before block A .

The dependency constraints when encoding a macroblock or a video frame can then be derived by merging that of each block. Figures 2(b) and 4(c) show the dependency constraints of macroblock encoding and frame encoding in AVS respectively, and Figures 3(b) and 4(b) depict that of H.264. As suggested by the figures, the dependency relationships form directed acyclic graphs (DAG)² [14]. The figures also suggest it is not possible to parallelize the RD cost computation of the four constituent blocks of the same macroblock in AVS. On the other hand, it may be possible to compute in parallel RD costs of the blocks from *different* macroblocks. Likewise, while it is not possible to parallelize the RD cost computation of the four constituent 4×4 blocks of the same 8×8 block in H.264, it may be possible to process in parallel the 4×4 blocks from different 8×8 blocks.

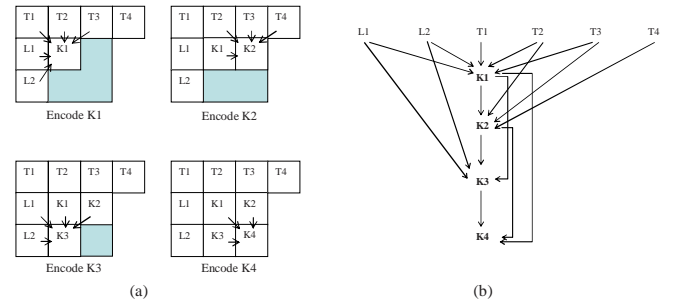


Fig. 2. (a) Dependency between the four 8×8 blocks ($K1, K2, K3, K4$) in the current macroblock and their spatially adjacent neighbor blocks ($T1, T2, T3, T4, L1, L2$), in AVS intra prediction. (b) Dependency constraints in encoding the four 8×8 blocks ($K1, K2, K3, K4$) of the current macroblock arising from AVS intra prediction.

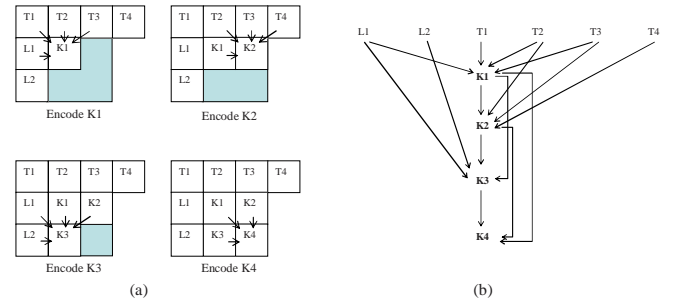


Fig. 3. (a) Dependency between the four 4×4 blocks ($K1, K2, K3, K4$) in the current 8×8 block and their spatially adjacent neighbor blocks ($T1, T2, T3, T4, L1, L2$), in H.264 intra prediction. (b) Dependency constraints in encoding the four 4×4 blocks ($K1, K2, K3, K4$) of the current 8×8 block arising from H.264 intra prediction. Note that these are similar to AVS except the dependency between $K1$ and $L2$ does not exist in H.264.

3.2. Greedy-based Block Encoding Order

Based on the discussed DAG representation of dependency, we propose a greedy-based block encoding order in this section to facilitate parallel intra mode decision. In particular, when compressing a video frame (with intra coding), instead of encoding each block one

²The graphs should be acyclic; otherwise, circular dependency between blocks would occur, and no proper encoding would be feasible.

by one following the conventional raster scan order, we propose to encode the blocks from different macroblocks in parallel subject to the dependency constraints in intra prediction, so that maximum parallelization can be achieved in computing the RD costs at the GPU. In the proposed encoding ordering, we encode, at each iteration, *all* the blocks of which the parent blocks have been encoded, i.e., we encode those blocks of which all the reference reconstructed pixels are available. This is illustrated in Figure 4(c) for AVS standard. In the first four iterations, we encode sequentially blocks A_1, A_2, A_3 and A_4 starting from the top-left most block A_1 (i.e., the root block). Then we encode in the 5-th iteration two blocks B_1 and D_1 in parallel from different macroblocks. This is feasible as their parent blocks (A_2, A_4 for B_1 , A_3, A_4 for D_1) have already been encoded. The encoding proceeds iteratively in this manner, and in each iteration those child blocks which *all* their parent blocks have been encoded will be selected for encoding. Figure 5(a) depicts this encoding ordering within an image frame.

Figures 4(b) and 5(b) depict our proposed encoding order to facilitate parallel H.264 4×4 block intra mode decision.

We remark that while we propose to encode the video blocks in the greedy-based order, the output bitstream will be organized conforming to the standards, i.e., bits are arranged according to the raster scan ordering of video blocks. Therefore, the output bitstream is standard compliant. The buffering and delay overheads of the proposed ordering are limited to that of encoding a single frame, and should be reasonable in PC or game consoles environments.

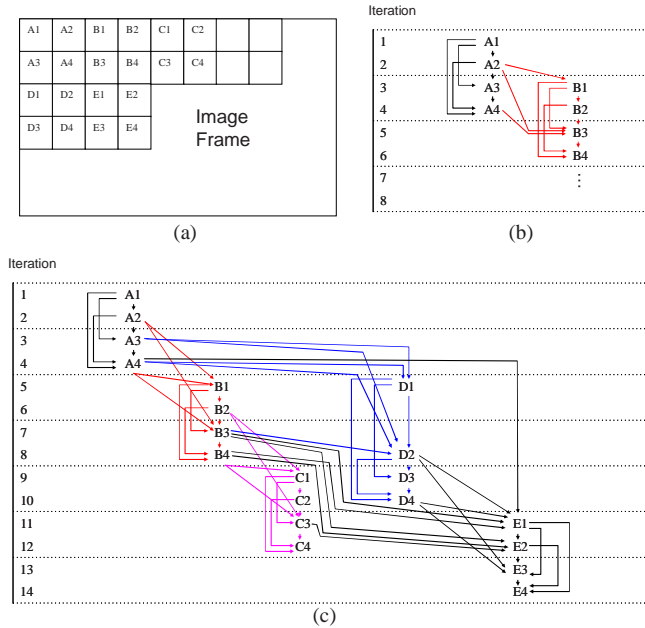


Fig. 4. (a) Notations for dependency graphs: each block corresponds to an 8×8 block in the case of AVS, or a 4×4 block in the case of H.264. (b) Dependency graph when encoding an image frame in H.264 standard. Each node represents an 4×4 block (See Figure 4(a) for notations). (c) Dependency graph when encoding an image frame in AVS standard. Each node represents an 8×8 block. The graph is processed following the proposed greedy-based processing order, and also shown in the figure is the iteration when each block is processed. Note that a video block will be scheduled for processing *immediately* after all its parents have been processed.

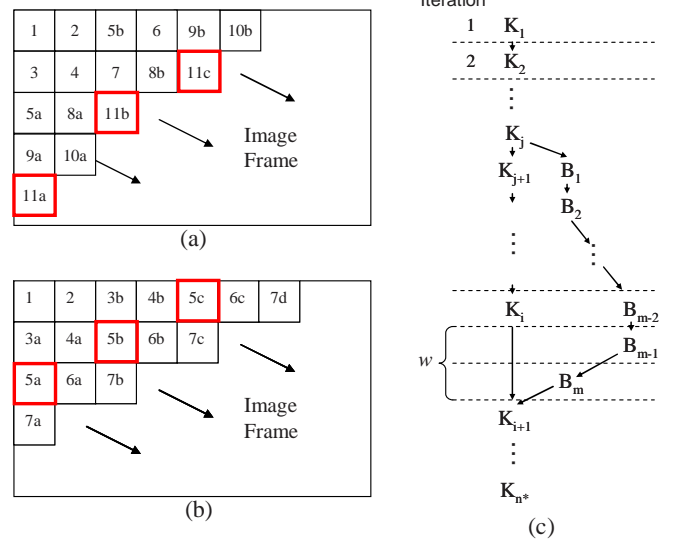


Fig. 5. (a) Our proposed greedy-based block encoding order for AVS encoding. Each square represents an 8×8 video block. Blocks with the same number (e.g., 11a, 11b, 11c) are to be processed in parallel in intra mode decision. (b) Our proposed block encoding order, for H.264 encoding. Each square represents a 4×4 video block. (c) Proof of Lemma 1.

3.3. Optimality

In this section we present the proof that the greedy-based encoding order is optimal, i.e., under the constraints imposed by AVS/H.264 intra prediction, and among all encoding orders obeying the constraints, the greedy-based encoding order requires the minimum number of iterations to process all the blocks (i.e., determine all the optimal intra mode decisions).

Definition. We use \mathcal{G} to denote the DAG representation of dependency constraints when encoding a video frame (i.e., Figure 4(c) for AVS, where each node corresponds to a video block). We define a *path* \mathcal{P} as a sequence of blocks $\{K_1, K_2, \dots, K_n\}$, where K_i is the parent node of K_{i+1} [14]. The length of \mathcal{P} , $|\mathcal{P}|$, is n . Of all the possible paths in \mathcal{G} , we define the *bottleneck path(s)* \mathcal{P}^* as the longest path(s) in \mathcal{G} with length n^* . It is clear that for any order, processing the entire \mathcal{G} would require at least n^* iterations. We will prove that the greedy-based order can indeed process \mathcal{G} with n^* iterations.

We would like to remark that under the AVS/H.264 constraints all the paths of \mathcal{G} start out from a common *root block* (the top-left block of a frame, e.g., A_1 in Figure 4) and end with a common block (the bottom-right block of a frame). These can be verified by tracing the dependency relationships (e.g., Figure 2). These properties are used in the proof.

Lemma 1. The proposed greedy-based encoding order can process all bottleneck path(s) \mathcal{P}^* with exactly n^* iterations³.

Proof of Lemma 1. Suppose to the contrary that the greedy-based order requires more than n^* iterations to process $\mathcal{P}^* = \{K_1, K_2, \dots, K_{n^*}\}$. Then there must exist at least one *processing gap* of length w iterations, $w > 0$, in which \mathcal{P}^* is not being processed, say between K_i and K_{i+1} (See Figure 4(c) for the gap between D_1 and D_2 of two iterations as an example). There would exist, however, an immediate parent block B_m of K_{i+1} ,

³We remark that the result assumes there are enough processing cores to support multiple parallel computation threads, and any delay would be solely caused by inter-blocks dependency. Since GPUs with hundred cores are common nowadays [4], the assumption is reasonable.

being processed immediately before K_{i+1} and belonging to another path (Figure 5(c)). This is true because the greedy-based order would process a video block *immediately* after all its parents have been processed, and therefore any block (except the root block) would have at least one parent block being processed in the immediately preceding iteration. Likewise, there would be another immediate parent block B_{m-1} processed right before B_m . Continuing with such backtracking eventually one would reach some block K_j in \mathcal{P}^* (one may reach \mathcal{P}^* at the root block, K_1 , the latest). The sub-path $\mathcal{P}_1 = \{K_j, B_1, B_2, \dots, B_m, K_{i+1}\}$ having no processing gap would thus be longer than the sub-path $\mathcal{P}_0 = \{K_j, K_{j+1}, \dots, K_i, K_{i+1}\}$ by at least w blocks. Therefore, the path $\{K_1, K_2, \dots, K_j, B_1, \dots, B_m, K_{i+1}, \dots, K_{n^*}\}$ formed by replacing \mathcal{P}_0 by \mathcal{P}_1 in \mathcal{P}^* would be longer than \mathcal{P}^* , contradicting that \mathcal{P}^* is the longest path in \mathcal{G} . \square

Theorem 1. The proposed greedy-based order can process all the video blocks in a frame in n^* iterations, where n^* is the length of the longest path in the DAG \mathcal{G} representing the dependency constraints imposed by AVS/H.264 intra prediction.

Proof of Theorem 1. The last block of a bottleneck path $\mathcal{P}^* = \{K_1, K_2, \dots, K_{n^*}\}$, K_{n^*} , would be processed in the n^* -th iteration by Lemma 1. Since all the paths would also end in K_{n^*} , all the blocks in \mathcal{G} could be processed with n^* iterations. \square

4. EXPERIMENTAL RESULTS

Experiments are conducted to evaluate the performance of the GPU-based RD optimized intra mode decision using the proposed encoding orders. We use a PC equipped with one GeForce 8800 GTS PCIe graphics card with 96 stream processors [4], and an Intel Pentium 4 3.2 GHz processor with 1 GB DDR2 memory, as the simulation platform. We use OpenGL API and programming language Cg to program the GPU.

We conduct experiments using the H.264 JM 14.0 reference software, with the RD cost computed as in (1). Table 1 shows the speedups of the proposed order compared with the conventional raster scan order on GPU (2nd column). The speedups are the ratios of GPU running time using the raster scan order to that using our proposed encoding order. As shown in the table, more than eighty times speedup can be achieved by the proposed order. Table 1 also shows the speedups of the proposed order compared with the strategy proposed in [9] running on GPU (3rd column). Note that [9] exploits parallelism only within a macroblock, while our work exploits parallelism in the whole frame to better harness the massive parallel capability of GPU. Table 1 fourth column compares our proposed parallel solution running on the GPU with the raster scan based intra prediction running on the CPU, in H.264 encoding. Note that in our platform the CPU runs at a higher clock rate, and CPU computation does not incur any CPU-GPU data transfer overhead, while GPU offers parallel processing opportunity. The results suggest GPU-based intra prediction using our proposed encoding order compares favorably to CPU-based intra prediction. Table 1 fifth column compares our proposed parallel solution running on the GPU with the raster scan based intra prediction running on the CPU, in AVS encoding and using (2) to compute the RD cost. The results also suggest the GPU-based intra prediction compares favorably to the CPU counterpart.

5. CONCLUSIONS

We have discussed our proposed GPU-based RD optimized intra mode decision. Based on the dependency analysis of intra mode decision, we proposed to encode the video blocks following greedy orders, leading to highly parallel RD cost computations. Simulation

Table 1. Speedup performance of the proposed parallel intra prediction in H.264 and AVS encoding ($QP = 28$): (i) compared with GPU implementation using conventional raster scan order, i.e., sequential, in H.264 intra prediction [2nd column]; (ii) compared with GPU implementation using [9] in H.264 intra prediction [3rd column]; (iii) compared with implementation on CPU in H.264 intra prediction [4th column]; (iv) compared with implementation on CPU in AVS intra prediction [5th column]. Note that the speedups are similar for different QP settings [12].

Sequence	Speedup with respect to:			
	Sequential	[9]	CPU (H.264)	CPU (AVS)
1280 × 720:				
crew	61.34	38.35	1.38	1.54
night	60.81	38.01	1.49	1.64
city	60.94	38.10	1.48	1.54
1920 × 1080:				
blue_sky	83.60	52.25	1.90	1.68
riverbed	84.07	52.55	1.93	1.68
station	83.96	52.47	1.89	1.69

results suggested the proposed GPU-based solution compares favorably to one using the single-core CPU. Since intra frame encoding would spend a significant portion of computation on RD optimized intra mode decisions, our strategy can potentially reduce the execution time of encoding intra frames significantly. Our future work includes investigating the optimal partition of tasks between CPU and GPU in intra frame encoding, taking into account system constraints such as the bandwidth between GPU and main memory.

6. REFERENCES

- [1] T. Wiegand, "Joint final committee draft for joint video specification H.264," Tech. Rep. JVT-D157, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, 2002.
- [2] L. Yu, F. Yi, J. Dong, and C. Zhang, "Overview of AVS-Video: tools, performance and complexity," in *Proc. Visual Communications and Image Processing (VCIP)*, 2005.
- [3] C.-L. Yang, L.-M. Po, and W.-H. Lam, "A fast H.264 intra prediction algorithm using macroblock properties," in *Proc. Int'l Conf. Image Processing (ICIP)*, 2004.
- [4] NVIDIA, "NVIDIA GeForce 8800 architecture technical brief," Tech. Rep., NVIDIA, 2006.
- [5] G. Shen, G.P. Gao, S. Li, H.Y. Shum, and Y.Q. Zhang, "Accelerate video decoding with generic GPU," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, pp. 685 – 693, 2005.
- [6] M.C. Kung, O. Au, P. Wong, and C.H. Liu, "Block based parallel motion estimation using programmable graphics hardware," in *Proc. International Conference on Audio, Language and Image Processing*, 2008.
- [7] G. Jin and H.-J. Lee, "A parallel and pipelined execution of H.264/AVC intra prediction," in *Proc. IEEE International Conference on Computer and Information Technology*, 2006.
- [8] W. Lee, S. Lee, and J. Kim, "Pipelined intra prediction using shuffled encoding order for H.264/AVC," in *Proc. IEEE Region 10 Conference (TENCON)*, 2006.
- [9] Ki-Won Yoo and Hyung-Ho Kim, "Intra prediction method and apparatus," *U.S. Patent Pub. No. US 2005/0089094*, 2005.
- [10] M.C. Kung, O. Au, P. Wong, and C.H. Liu, "Intra frame encoding using programmable graphics hardware," in *Proc. Pacific Rim Conference on Multimedia (PCM)*, 2007.
- [11] Todd C. Mowry, "List Scheduling," Lecture Notes in Carnegie Mellon University CS745: Optimizing Compilers.
- [12] N.-M. Cheung, O. Au, M.C. Kung, H.W. Wong, and C.H. Liu, "Highly parallel rate-distortion optimized intra mode decision on multi-core graphics processors," accepted to *IEEE Trans. Circuits and Systems for Video Technology - Special Issue on Algorithm/Architecture Co-Exploration of Visual Computing*, 2009.
- [13] Y.-K. Chen, W. Li, J. Li, and T. Wang, "Novel parallel Hough transform on multi-core processors," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.
- [14] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1985.