Parallel Streaming Intra Prediction for Full HD H.264 Encoding

Ju Ren, Yi He, Huayou Su, Mei Wen, Nan Wu, Chuntyuan Zhang
Computer School
National University of Defense Technology
Changsha, China
se7en.rj@gmail.com

Abstract—Intra prediction is the most important intensive computing component in H.264 intra frame coder. Its high computational costs give huge pressure to most current embedded programmable processors, especially in real-time HD H.264 video encoding. Stream processing model, an emerging parallel processing model supported by GPUs and most programmable processors, bridges the gap between flexible programmable processors and high-performance special-purpose processors. This paper presents parallel streaming intra prediction on programmable processors. Two methods of 7-steps block parallel and macro block parallel are designed to deal with the restrictions of parallel. Experiment results of full HD H.264 encoding show that our parallel streaming intra prediction methods accomplish obvious speedup (more than 5 times) on different programmable processors. Moreover, the entire streaming H.264 encoder with our parallel intra prediction method achieves real-time performance for encoding full HD H.264 video sequence.

Keywords—intra prediction; stream; full HD; real-time; programmable processor

I. INTRODUCTION

In recent years, consumer electronics have developed quickly, manifested in that embedded processors and applications are gradually part of people’s daily life. Multimedia application, especially real-time video coding application, is one of the most common tasks on various embedded processors (digital camera, MP4 player, Mobile phone, videogame consoles, etc.). On one hand, people’s needs promote video resolution of embedded multimedia applications to a high definition, during which compression rate in video coding is the foremost indicator. On the other hand, most of embedded programmable processors (e.g. Intel ATOM) are developed for various applications including HD video coding, during which flexibility of video encoder is the foremost factor. Thus, known for its high compression rate, high visual quality and flexibility, H.264 standard [1] is widely adopted by most embedded video coding applications.

However, high compression rate and high image quality of H.264 come at the price of significantly increased computational complexity due to the new techniques. Intra prediction is one of the new techniques. Lots of researchers have been seeking techniques either on software or hardware [2, 3] to accelerate intra prediction. But software algorithm in serial manner cannot meet real-time HD requirements on embedded programmable processors, while dedicated ASIC hardware still has limitations of long development time and inflexibility.

At present, the multi-core embedded programmable processors have been developed (Dual core ATOM, CELL in PlayStation 3, etc.). And most researches of fast intra prediction algorithm do not break the limits of serial computing characteristics. Therefore, we attempt to develop a software parallel method fit for multi-core programmable processors in order to accelerate the inter prediction.

Stream processing model [4] is an emerging parallel processing model supported by GPUs and programmable processors. By exploiting the application’s ILP, DLP, TLP efficiently, stream processing model bridges the gap between flexible general-purpose processors and high-performance special-purpose processors. It shows surprising efficiency on many compute-intensive domains especially for media processing [5]. Studies in [6] have parallelized many key components of H.264 on stream processing model with satisfactory results. This may provide a feasible approach to parallelize the intra prediction on stream processing.

This paper presents software parallel intra prediction based on stream processing model. First we make a profile of intra prediction to analyze its inherent characteristics and restrictions of parallel. Then we design two methods in order to eliminate the dependency: 7-steps block parallel method and macro block parallel method. At last, the two proposed intra prediction methods are applied in two different programmable processors (GPU and stream processor SP16). The results show that our intra prediction methods have obvious speedup (more than 5x) over the serial code on GPU and stream processor. Furthermore, the results also indicate that the performances of two parallel streaming intra predictions can satisfy the real-time requirements of full HD 1080p H.264 encoding on stream processor.

The rest of this paper is organized as follows. Section 2 presents an overview of stream processing. In section 3, we give a profile and analysis for parallelizing intra prediction. Section 4 describes the novel parallel streaming intra prediction method. Evaluation results are given in Section 5. The paper is summarized in Section 6.

Thanks to the sponsors of China NSFC No.60703073, No.60903041.
II. STREAM PROCESSING MODEL

Stream is an ordered set of data of an arbitrary data type. Stream processing model takes streams as primitive processing elements and decomposes applications into a series of computational kernels connected by streams [7]. Operations in the stream processing model include streamgather and streamscatter, and computation in the form of kernel. Kernel, operating on one or more streams, is an individual computation process. Each kernel reads data from its input streams, then executes operations on stream data and finally writes data into its output streams.

![Figure 1. Stream processing model](image)

Transforming a regular program to a stream program requires analyzing the dataflow through the desired algorithm, dividing the data into streams and dividing the computation into kernels, which is called streaming [8]. As shown in figure 1, application is decomposed by a collection of streams passing through a series of kernels in stream processing model.

Nowadays, stream processing model is widely supported by most programmable processors, including GPU and multi-core processors. That means stream processing model could exploit locality and parallelism of the applications on programmable processors [9].

![Figure 2. Stream architecture](image)

Figure 2 is the abstract architecture of the stream processors. Stream processor is composed of PPU (parallel process unit) array with inter PPU switch, PPU controller, shared on-chip memory, and off-chip memory system. During stream program execution, data are organized into streams which are transferred from off-chip memory to on-chip memory. Then PPUs execute kernels in SIMD manner, for which multiple stream elements in different PPUs are processed by the same instruction simultaneously. The output streams from PPUs are stored in the on-chip memory and finally transferred to off-chip memory. In the whole executing procedure, all operations in kernels are executed in parallel. Therefore, the architecture brings parallel into full play in stream processing by exploiting DLP, ILP and TLP.

III. PARALLELISM ANALYSIS

A. Intra Prediction Profile

Intra prediction of H.264 has multiple prediction modes including 4x4 luma prediction, 16x16 luma prediction and 8x8 chroma prediction. There are 9 prediction modes for 4x4 luma prediction as shown in figure 3. For 16x16 luma prediction, there are 4 prediction modes as depicted in figure 4. Four modes of 8x8 chroma prediction are similar to 16x16 luma prediction.

![Figure 3. Nine 4x4 prediction modes](image)

![Figure 4. Four 16x16 prediction modes](image)

Intra prediction is processed in a serial manner in conventional H.264 encoders. For each current frame, macro
blocks are processed in a raster scan order. In a macro block, 16 blocks are processed in a zig-zag order as shown in figure 5. The serial processing manner makes intra prediction a time-consuming part of H.264.

![Serial 4x4 intra prediction order](image)

Figure 5. Serial 4x4 intra prediction order

In our previous studies, H.264 encoder has been executed in serial mode on CPU. The breakdown of different components in H.264 encoder is depicted in figure 6. Because intra prediction and “transform and quantization” are tightly combined, we put them together in one part. The portion of intra prediction and transform and quantization accounts for 42%. Thus, we attempt to design a parallel intra prediction for accelerating the encoding performance.

![Execution time breakdown for each part in H.264](image)

Figure 6. Execution time breakdown for each part in H.264

### B. Restrictions of Parallel

Before parallelizing intra prediction, the restrictions of parallel should be considered first. In fact, the intra prediction procedures in all 16 blocks are identical. It seems that different blocks can be processed at the same time. But data dependencies of intra prediction make it hard to be parallelized. There are two kinds of data dependencies in intra prediction. One is the spatial data dependency, and the other is temporal data dependency.

![Data dependencies between blocks](image)

Figure 7. Data dependencies between blocks

Spatial data dependency is derived from pixels of the neighboring blocks. As shown in figure 3 and figure 4, luma and chroma prediction procedure needs to use the boundary pixels of the neighboring blocks. Figure 7 illustrates the data dependencies between blocks. Apparently, prediction of a block depends on the upper left block (B0), the above block (B1), the upper right block (B2) and the left block (B3). Block dependency also causes dependency between macro blocks, shown in figure 7.

Temporal data dependency is derived from the reconstruction procedure. Reference data in figure 6 are not the original pixels in the neighboring blocks but reconstruction data computed by residual transform and quantization, inverse quantization and inverse transform, and reconstruction procedure.

### IV. PARALLEL STREAMING INTRA PREDICTION

Two parallel methods to eliminate restrictions in section 3 are described in section 4A and 4B.

#### A. 7-steps Block Parallel Intra Prediction

In order to parallelize Intra Prediction, an appropriate data processing granularity should be chosen first. Different processing granularities including frame, slice, macro block, and block are discussed as follows with the assumption that the number of PPU is N.

Slice or Frame parallel. Because intra prediction does not go beyond the slice boundary in H.264 standard, slices or frames can be executed in parallel. But this method makes long input streams which would cause huge memory pressure for on-chip memory. And for most of embedded processors, on-chip memory is very expensive resource.

Macro block parallel. In this manner, one 16x16 macro block is processed by one PPU. Unfortunately, dependencies between neighboring macro blocks mentioned in section 3 make this method not work.

Block parallel. We choose block parallel method for our streaming model of intra prediction. In this manner, one 4x4 block is processed by one PPU. It seems that dependencies between blocks make this method unfeasible too. In serial intra prediction encoder, there are 16 steps for 4x4 intra prediction of one macro block as indicated in figure 7. But by analyzing the dependencies of each block in one macro block, a number of blocks can be processed simultaneously. For example, block 4 and block 2 in figure 5 can be processed at the same time since the reference data of block 4 is only in block 1. As shown in figure 8(a), processing steps of one macro block can be reduced from 16 to 10. In this manner, at least two PPUs should be used, because two blocks need to be processed simultaneously in each step from step 2 to step 7.

Furthermore, the 10 steps block parallel intra prediction can be reduced into 7 steps. It can be seen that the pixels E-H are used only for two prediction modes: mode 3 and mode 7. And experiments show that executing prediction without 2 modes increases the bit rate by less than 2% while decreases PSNR value less than 0.02 [10]. Thus we optimize 10-steps method to 7-steps block parallel intra prediction method (7BPIP) by eliminating the 2 modes. Figure 8(b) illustrates the 7 prediction modes procedure. In this method, dependency of top-right
block is eliminated, and the largest parallel block numbers increase from 2 to 4 (4 blocks can be processed simultaneously in step 3). Therefore, only 7 steps are needed and 4 PPUs can be used in parallel.

Figure 8. Ten and seven steps block parallel prediction

For 16x16 intra prediction, block parallel method is adopted in vertical, horizontal and DC prediction mode, because process of blocks in these modes is the same. 8x8 chroma intra prediction is similar to 16x16 intra prediction.

B. Macro Block Parallel Intra Prediction

If only spatial data dependency and partial temporal data dependency are considered, macro block parallel intra prediction method could be used.

In this method, one 16x16 macro block is processed by one PPU, while N macroblocks in N PPUs are processed in parallel. Figure 9 illustrates the macro block parallel intra prediction (MBPIP) with the assumption of N=4. The reference data of a macro block are H data and V data shown in figure. H data have been reconstructed for predicting macro blocks in the next row while V data have not. Thus, we have to choose the unreconstructed V data. Results show that, for high definition sequences, the quality loss is negligible [11]. For instance, macro block 5 is current macro block, and H data comes from reconstructed pixels of top macro block 1 and V data is from original pixels of left macro block 4. One advantage of this method is that dependency between 4x4 blocks of one macro block is eliminated because 16 blocks in one macro block are processed in serial manner.

C. Optimizing Implementation of Parallel Intra Prediction

We implement the proposed intra prediction methods on two platforms of Geforce 8800GTX GPU and STORM SP16-G220 for 1080p HD encoding. There are 128 stream processors in GPU and 16 computing lanes in SP16 for parallel processing.

The key points of 7-steps block parallel intra prediction method implementations are provided as follows.

Figure 10. Location of luma and chroma stream data

Prepare data into streams: Streams in the implementation include input streams, output streams and intermediate streams. Input streams are organized by input YUV data, while output streams are reconstructed data and quantized transform coefficients. According to the 7BPIP method, data of one block should be loaded into one PPU. Since there are 16 PPUs in GPU and SP16, 16 blocks of a macro block accurately locate to 16 PPUs. The initial input streams are composed of more than one macro blocks, and an inner iteration in the kernel process one macro block. The number of macro blocks in the input stream is decided by the capacity of on-chip memory and input frame resolution. In our implementation, the number is 120 (1920x1080 frame consists 120 macro blocks in a row). Thus each kernel has 120 inner iterations for processing all macro blocks in a row. Figure 10 illustrates the location of luma and chroma stream data on 16 PPUs. For chroma prediction, U and V data are connected into one input stream.

Kernels: The flowchart of kernels and streams is shown in figure 11. For each loop of intra prediction encoder, kernels complete one macro block intra prediction. Moreover, the transform and quantization are also implemented within our intra prediction encoder since the two components are close related. Transform and quantization can be easily parallelized, because there is no dependency in it. k_TQR4 and k_TQR16 in figure 11 are kernels of transform and quantization.

For macro block parallel intra prediction method, each PPU loads one macro block data. The reference data of V are got by communicating the original pixels between neighboring PPUs.
V. RESULTS AND ANALYSIS

A. Experiment Setups

Our target application is full HD H.264 encoding on baseline profile. Table 1 lists configurations of three different full HD video sequence. Blue_sky has the rotating perspective and large area with gradual changed color. In Station2, the background area has delicate details. Rush_hour has lots of moving cars in the fog. The benchmarks with different characters can evaluate our methods in the aspects of accuracy and performance.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Core Number</th>
<th>Frequency</th>
<th>Memory</th>
<th>On-chip Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>STORM SP16 G220</td>
<td>1 MIPS</td>
<td>330MHz</td>
<td>1GB</td>
<td>256K</td>
</tr>
<tr>
<td>NVIDIA GeForce 8800GTX</td>
<td>16 stream</td>
<td>1.35GHz</td>
<td>768M</td>
<td>256K</td>
</tr>
<tr>
<td>X86 Core2 E8200</td>
<td>2 cores</td>
<td>2.67GHz</td>
<td>4GB</td>
<td>128K</td>
</tr>
</tbody>
</table>

B. Performance Analysis

Table 3 lists execution time and speedup of reference and parallel methods on the two platforms. It is noticeable that the reference code execution time in the fourth column is not the time on GPU or stream processor, but the time CPU performed. The speedups are the ratios of reference code running time on CPU to our proposed parallel prediction running time on GPU or stream processor.

As shown in table 3, all the three video sequence have similar speedup over the reference code. 7BPIP gains 5.1 times speedup on GPU platform, while it gains 6.3 times speedup on stream processor platform. And MBPIP gets more than 7 times speedup on stream processor platform. The results suggest that the two proposed methods can harness the parallel capability of GPU and stream processor. Another reason for the outstanding speedup is that stream processing model can exploit the producer-consumer locality efficiently. Most of intermediate data streams stay on the on-chip memory and are used by subsequent kernels as shown in figure 12.

The speedup of MBPIP is higher than that of 7BPIP on the stream processor platform. That is because MBPIP has more parallel blocks than 7BPIP. There are 16 blocks processed simultaneously in 16x16 luma prediction of 7BPIP and MBPIP. But the number of parallel blocks in 7BPIP 4x4 luma prediction is 1 to 4 according to the different steps, while that

For GPU platform configuration, we use a PC with one Intel Core2 E8200 CPU and one GeForce 8800 GTX graphics card as the experiment platform. For stream processor platform configuration, we use SPI STORM SP16 G220 stream processor. Table 2 lists configurations of GPU, SP16 stream processors and CPU. Both of our two parallel streaming methods are executed on the stream processor platform. 7-steps block parallel intra prediction is implemented on the GPU platform. In addition, reference code of x264[12] is executed on the CPU for comparison. CUDA [13] and SPI stream programming language [14] are used for programming on the two platforms respectively. In our implementation of reference code and two proposed parallel intra predictions, low complexity mode is chosen for computing Lagrangian costs.

<table>
<thead>
<tr>
<th>Platform</th>
<th>video sequence</th>
<th>data time</th>
<th>reference code time</th>
<th>7BPIP</th>
<th>MBPIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>Blue sky</td>
<td>16.20s</td>
<td>3.15s</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Station2</td>
<td>23.41s</td>
<td>4.68s</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Rush_hour</td>
<td>38.86s</td>
<td>7.63s</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SP16</td>
<td>Blue sky</td>
<td>16.20s</td>
<td>2.56s</td>
<td>2.18x</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Station2</td>
<td>23.41s</td>
<td>3.76s</td>
<td>3.19x</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>Rush_hour</td>
<td>38.86s</td>
<td>6.18s</td>
<td>5.15x</td>
<td>N</td>
</tr>
</tbody>
</table>

As shown in table 3, all the three video sequence have similar speedup over the reference code. 7BPIP gains 5.1 times speedup on GPU platform, while it gains 6.3 times speedup on stream processor platform. And MBPIP gets more than 7 times speedup on stream processor platform. The results suggest that the two proposed methods can harness the parallel capability of GPU and stream processor. Another reason for the outstanding speedup is that stream processing model can exploit the producer-consumer locality efficiently. Most of intermediate data streams stay on the on-chip memory and are used by subsequent kernels as shown in figure 12.

The speedup of MBPIP is higher than that of 7BPIP on the stream processor platform. That is because MBPIP has more parallel blocks than 7BPIP. There are 16 blocks processed simultaneously in 16x16 luma prediction of 7BPIP and MBPIP. But the number of parallel blocks in 7BPIP 4x4 luma prediction is 1 to 4 according to the different steps, while that
prediction. Macro block parallel intra prediction increases designed on the basis of dependency analysis of intra processing order, 7-step block parallel intra prediction is macro block parallel block intra prediction. By using different methods of 7-step block parallel block intra prediction and different macro blocks.

number in MBPIP is always 16 because they come from different macro blocks.

We also implement the entire parallel streaming H.264 encoder on STORM SP16 by applying our parallel intra predictions in it. Table 4 and table 5 illustrate the performance and compression rate of the two methods. The results of the third column in the tables indicate that both two proposed parallel streaming intra prediction methods can satisfy the real-time requirements of full HD H.264 encoding with a little higher compression rates. The results also show that MBPIP achieves higher performance than 7BPIP, while compression rate of 7BPIP is higher than that of MBPIP.

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>Time (s)</th>
<th>FPS</th>
<th>Compression rate of reference</th>
<th>Compression rate of MBPIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue sky</td>
<td>7.4s</td>
<td>29.3</td>
<td>72:1</td>
<td>71:1</td>
</tr>
<tr>
<td>Station2</td>
<td>10.8s</td>
<td>29.0</td>
<td>202:1</td>
<td>199:1</td>
</tr>
<tr>
<td>Rush hour</td>
<td>17.3s</td>
<td>28.9</td>
<td>113:1</td>
<td>112:1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Video sequence</th>
<th>Time (s)</th>
<th>FPS</th>
<th>Compression rate of reference</th>
<th>Compression rate of MBPIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue sky</td>
<td>7.1s</td>
<td>30.6</td>
<td>72:1</td>
<td>70:1</td>
</tr>
<tr>
<td>Station2</td>
<td>10.3s</td>
<td>30.4</td>
<td>202:1</td>
<td>196:1</td>
</tr>
<tr>
<td>Rush hour</td>
<td>16.6s</td>
<td>30.1</td>
<td>113:1</td>
<td>110:1</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

This paper discussed two parallel stream intra prediction methods of 7-step block parallel block intra prediction and macro block parallel block intra prediction. By using different processing order, 7-step block parallel intra prediction is designed on the basis of dependency analysis of intra prediction. Macro block parallel intra prediction increases parallel granularity by ignoring data dependency between macro blocks. The experimental results show that our proposed methods can achieve more than 5 times speedup over serial reference code on GPU and stream processors. Furthermore, the performances of two parallel streaming intra predictions can satisfy the real-time requirements of full HD 1080p H.264 encoding on stream processor. The results also suggest that our intra predictions have a little impact to the coding performance of less than 3% increase of compression rate.

The high performance is achieved by designing the intra prediction into stream processing model which can efficiently exploit parallelism and locality in the application. Without additional requirement for any special dedicate ASIC design or algorithm, the parallel streaming methods can be applied on most of programmable processors. Moreover, the proposed parallel methods can fit for video compression of higher definition than 1920x1080 by increasing kernel loops.

REFERENCES