PAPER Special Section on Parallel and Distributed Computing and Networking

Parallelization of Computing-Intensive Tasks of the H.264 High Profile Decoding Algorithm on a Reconfigurable Multimedia System

Tongsheng GENG[†a)], **Leibo LIU**[†], ***Nonmembers***, **Shouyi YIN**[†], ***Member***, **Min ZHU**[†], ***and*** **Shaojun WEI**[†], ***Nonmembers***

SUMMARY This paper proposes approaches to perform HW/SW (Hardware/Software) partition and parallelization of computing-intensive tasks of the H.264 HiP (High Profile) decoding algorithm on an embedded coarse-grained reconfigurable multimedia system, called REMUS (REconfigurable MUltimedia System). Several techniques, such as MB (MacroBlock) based parallelization, unfixed sub-block operation etc., are utilized to speed up the decoding process, satisfying the requirements of real-time and high quality H.264 applications. Tests show that the execution performance of MC (Motion Compensation), deblocking, and IDCT-IQ (Inverse Discrete Cosine Transform–Inverse Quantization) on REMUS is improved by 60%, 73%, 88.5% in the typical case and 60%, 69%, 88.5% in the worst case, respectively compared with that on XPP PACT (a commercial reconfigurable processor). Compared with ASIC solutions, the performance of MC is improved by 70%, 74% in the typical and in the worst case, respectively, while those of Deblocking remain the same. As for IDCT_IQ, the performance is improved by 17% no matter in the typical or worst case. Relying on the proposed techniques, 1080p@30 fps of H.264 HiP@ Level 4 decoding could be achieved on REMUS when utilizing a 200 MHz working frequency.

key words: *H.264, reconfigurable multimedia system, parallelization computation, hardware/software partition*

1. Introduction

With the development of the Internet and wireless devices, interactive multimedia processing and high performance mobile computing are becoming more and more important. Currently, multi-standards (e.g. JPEG, MPEG-2, MPEG-4, H.263, H.264, etc.) exist in the media processing market, among which H.264, aiming at high-quality video content and a low bit rate, requires much more computation than most existing standards [1], [2]. It is impossible to meet the real-time requirements when performing H.264 HiP 1080p@30 fps decoding purely by GPP [3] (General Purpose Processor) and PDSP (Programmable Digital Signal Processor) [4]. To address this issue, GPPs and PDSPs are usually incorporated with hardware accelerators (i.e. ASIC modules) to speed up the computing-intensive tasks. However, the diversification and continuous evolution of multimedia algorithm standards and the seamless transition between various media processing algorithms pose higher flexibility requirements for this approach. Hardware design is characterized by a long design cycle and high cost, making it extremely difficult for hardware solutions to keep up with the evolution of applications. A novel approach, reconfigurable computing, has recently become the premier focus of both the academy and the industry, relying on its good trade-off property between GPPs' high flexibility and ASICs' high energy efficiency. Coarse-grained reconfigurable processors (XPP PACT [5], Morphosys [6], and ADRES [7]) have proved to be efficient solutions to speed up media processing applications.

A high concentration of computational complexity is a prominent feature of multimedia algorithms, whose tasks always consume the majority of execution time. In the H.264 HiP decoding algorithm, the tasks of MC, Deblocking, Intra prediction, IDCT-IQ, and entropy decoding, such as CAVLC (Context-Adaptive Variable Length Coding) and CABAC (Context-Adaptive Arithmetic Binary Coding), consume most of the execution time, in which MC, Deblocking, and IDCT-IQ together account for 76% of the total execution time, as illustrated in Fig. 1. These three tasks share the following properties in media processing algorithms: MB-based operations, an intrinsic regular dependence mechanism in inner MBs (all the data in one MB follow the same computing regulation), and an independence mechanism in different MBs (the data in different MBs is unrelated, but the operations of different MBs are almost identical). All of these features are suitable for reconfigurable computing, where functions are dynamically configured into a PEA (Processing Element Array) at run-time and carried out in parallel.

Current reconfigurable approaches, some [5], [6], [8] only focus on speeding up the computing-intensive tasks rather than speeding up the entire algorithm, while some only focus on reducing the power consumption by optimizing memory access scheme [9]. Normally, a typical task in reconfigurable computing can be divided into three sub-tasks: configuration, data I/O, and calculation. According to the statistics of this paper, the calculation sub-task only accounts for a minor portion of the whole task, as illustrated in Fig. 2. Furthermore, algorithm parallelism has not been analyzed and verified fully, either in these reconfigurable architectures or in ASIC solutions. For MC, XPP PACT, platform-MC [10], and ASIC-MC [11] all use a fixed block size (either 4×4 or 8×8) as the basic processing

Manuscript received February 3, 2010.
Manuscript revised May 24, 2010.

[†]The authors are with Institute of Microelectronics, Tsinghua University, Beijing, China.
a) E-mail: gts07@mails.tsinghua.edu.cn

DOI: 10.1587/transinf.E93.D.3223

unit, rather than a variable block size that can adapt to different block size partitions. Consequently, these platforms suffer from significant pressure of memory access. Moreover, sub-block parallelism is not available when utilizing a fixed size method.

For Deblocking, although sub-block parallelism is widely used on platforms such as XPP PACT and ASIC-Deblocking [16], the efficiency is heavily limited by data dependency residing in sub-blocks. For IDCT-IQ, XPP PACT and ASSP-IDCT [17] only focus on the parallelism of intra-blocks, while little attention is given to the parallelism of inter-blocks.

In this paper, parallelization of computing-intensive tasks of H.264 HiP decoding algorithms is implemented on REMUS. The proposed parallelization methods include: variable size block partition, unfixed sub-block operation, and sub-block parallelization for MC, MB-based parallelization for Deblocking and inter-blocks parallelization for IDCT, etc. The proposed methods fully exploit the parallelism of the computing-intensive algorithms, reduce redundant data transmission and memory access, and therefore greatly enhance execution efficiency. REMUS is a reconfigurable multi-processor SoC designed by our research group; it consists of two major components: a host processor (implemented by ARM1176JZ), which is in charge of the relatively complex judgment of branch and procedure control, and a PEA, which serves as a coprocessor responsible for intensive and regular calculation. Table 1 shows a comparison of some key characteristics between REMUS and the other popular reconfigurable systems. Tests show that, with the methods proposed in this paper, the performance of MC is improved by 60%, Deblocking by 73%, and IDCT-IQ by 88.5% in the typical case and by 60%, 69% and 88.5% in the worst case, compared with those of XPP PACT. Compared with some ASIC solutions [11], [16], [17], the performance of MC is improved by 70% in the typical case and 74% in the worst case, IDCT by 17% (both in the typical and worst case), while those of Deblocking are nearly the same. Relying on all these improvements, real-time decoding (1920×1080@30 fps) of H.264 HiP@ Level 4 could be realized on REMUS when exploiting a 200 MHz working frequency.

This paper analyzes the parallel characteristics of the typical H.264 algorithm, proposes optimization methods in Sect. 2, followed by verification of these methods on REMUS in Sect. 3. Conclusions are discussed in Sect. 4.

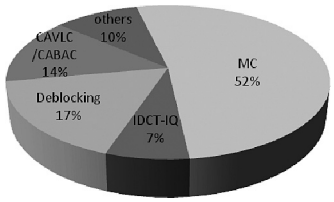


Fig. 1 Distribution of algorithms' computational complexity in H.264 HiP decoding (foreman).

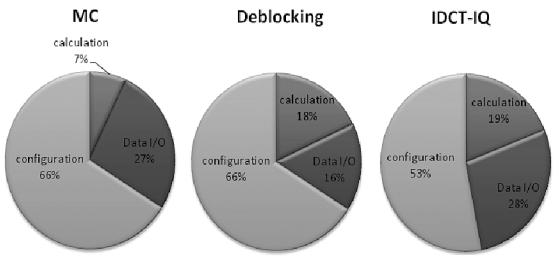


Fig. 2 Timing chart of different sub-tasks on a reconfigurable system.

Table 1 Comparison of characteristics of reconfigurable systems.

System	*Granularity*	*Programmability*	*Reconfiguration*	*Interface*	*Comp. Model*	*Application domain*
Splash 2	Fine-grained	Multiple	Static	Remote	Uniprocessor	Complex bitoriented computations
XPP PACT[5]	Coarse-grained	Multiple	Dynamic	Local	Uniprocessor& SIMD	computing-intensive, media application
Morphsys [6]	Coarse-grained	Multiple	Dynamic	Local	SIMD	Dataparallel, computing- intensive application
Garp[12]	Fine-grained	Multiple	Static	Local	Uniprocessor	Bit level image processing, cryptography
RaPiD[13]	Coarse-grained	Single	Mostly static	Remote	Pipelined Processor	Systolic arrays, regular, computing-intensive
DAPDNA [14]	Coarse-grained	Multiple	Dynamic	Remote	Uniprocessor	Computing-intensive
Warp [15]	Fine-grained	Single	Dynamic	Local	Uniprocessor	Undefined
REMUS	Coarse-grained	Multiple	Dynamic	Local	Pipelined Processor	computing-intensive, media & baseband application

2. Analysis and Optimization of Several Typical Algorithms

2.1 MC

(1) Parallelism Analysis of MC

In order to achieve high decoding quality and a low bit rate, MC in H.264 adopts more flexible and efficient technologies, such as TSVBS (tree structure variable block size), SPI (sub-pixel interpolation), and MRF (multiple reference frames) [2], [18]. However, these three technologies increase the complexity of computing, memory access, and control logic in the implementation of the H.264 decoder, which makes MC a major bottleneck in real-time mobile or high-resolution video applications. H.264 allows four basic MB partition modes and four different sub-block combination modes, as shown in Fig. 3 (a). As for the sub-pixel interpolation, the 6-tap filter is used for interpolating half-pixels of luma and the average of two integer/half-pixels for quarter-pixels of luma, while the bilinear interpolation filter is used for eighth-pixels of chroma [18]. The 6-tap filter requires that a total of $(M + 5) \times (N + 5)$ bytes of reference data must be loaded for interpolation of quarter-pixels in an $M \times N$ block of luma in the worst case, as shown in Fig. 3 (b). The memory bandwidth becomes the main problem for TSVBS and SPI. Because 4×4 is the minimum partition granularity of MBs, many designs take 4×4 as the basic unit to support eight different kinds of sub-block size. However, this method causes serious overhead on memory access bandwidth: the smaller the partition basis, the more redundant the reference data. Provided to predict a luminance block of 8×8 size, the reference data is $(8 + 5) \times (8 + 5) = 169$ bytes at the very least. If the calculation is based on the unit of 4×4 block, the reference data is $(4+5)\times(4+5)\times4 = 324$ bytes, with repetitive reference data as much as 155 bytes. Similarly, if the block size is 16×16, the number of reference data and repetitive data is 441 bytes and 855 bytes, respectively, with the repetitive data nearly two times the valid data. If the reference data can be read according to the partition size of MB, the efficiency will be improved greatly.

Generally speaking, the process of MC can be partitioned into two parts: computing the reference address and interpolation. The first part contains many judgments of branch and irregular computation, which is very difficult to schedule. The second part has the characteristics of intensive and regular calculation and relative data independency, making parallelism possible, discussed below in more detail:

(a) Regularity of calculation: Despite the tiny difference in interpolation of different fractional positions, the calculations are all based on the 6-tap filter. Furthermore, the steps of calculation are the same in each sub-block.

(b) Relative independency of data: In H.264, different sub-blocks may have different reference frames, so the calculation is independent among sub-blocks. Furthermore, even in one sub-block, the filtering of one row is independent from another. The same is true in column filtering.

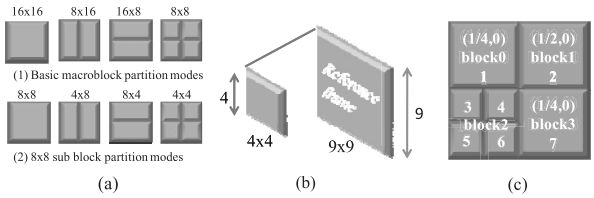


Fig. 3 (a) MB partition model, (b) 4×4 block and reference frame, (c) Luma 16×16 execute order.

(2) Optimization Methods of MC

According to the analysis in part (1), in the MC algorithm, the part of irregular computation can be implemented by software for efficient scheduling and the part of intensive computation can be implemented by hardware for acceleration. For the acceleration part, the technologies of variable block size, inter-block parallelism, and pipeline can be applied to improve efficiency and reduce memory access.

First, because the fixed block size (4×4 or 8×8) lacks flexibility, the method of variable block size will be taken to improve the utilization of data.

Second, the steps of calculation are fixed when the FPP (Fraction-Position Prediction) and the size of blocks are fixed. However, the calculation order of sub-blocks can be adjusted to improve efficiency, for example, putting sub-blocks owning same FPP together, not necessarily strictly following the order of the MB partition. In Fig. 3 (c), the original order of execution is from 1 to 7, but the FPPs of block0 and block3 are the same, so we can put the operation of block0 and block3 together. Through this method, the more complex of calculation (same FPPs of different sub-blocks), the faster of implement.

Third, for the reason that the calculation of different rows/columns is independent, the technology of pipeline can also be employed to improve efficiency significantly.

2.2 Deblocking

(1) Parallelism Analysis of Deblocking

In H.264, the Deblocking filter [19] is used to decrease blocking artifacts at block boundaries, which is caused by block-based IDCT in intra- and inter-frame prediction error coding, coarse IQ, and MC prediction filter. The high computation complexity caused by the high self-adaptive characteristic of the Deblocking algorithm makes it difficult to be implemented to meet the requirements of real-time. The coexistence of conditional checking and intensive calculation is a typical characteristic of Deblocking. Almost every pixel has to be loaded from memory for conditional checking, which leads to a massive memory access bandwidth and consumes a large number of clock cycles.

Although a lot of uncertainty exists, the operation of the Deblocking process can be divided into two steps: BS

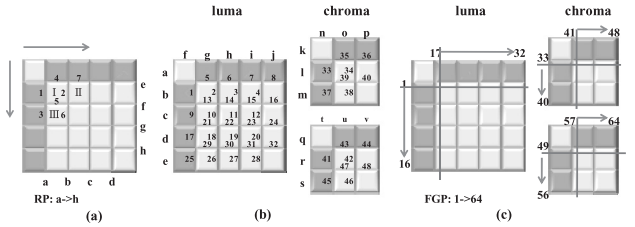


Fig. 4 (a) RP order, (b) SBP order, (c) FGP order.

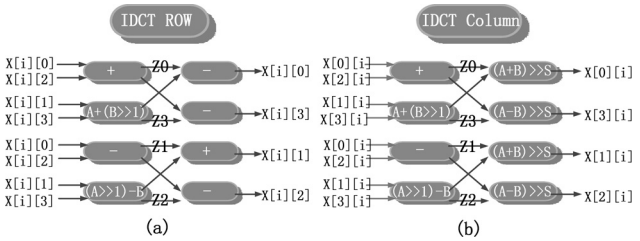


Fig. 5 (a) IDCT DFG for row, (b) IDCT DFG for column.

(Boundary Strength; to determine the filter strength) selection and filter implementation. For one sub-block, the order of the Deblocking filter is determined by data dependency. For example, the left boundary data updated by the filter will be used in the operation of right boundary filter, and the upper boundary data will be used for the lower boundary. As for one MB, horizontal filtering of vertical edges from left to right is performed first, followed by vertical filtering of the horizontal edges from top to bottom. Figure 4 (a) presents vertical and horizontal boundaries in one MB; the operation of the vertical/horizontal boundaries of every 4×4 block in each MB is the same. The parallelism is difficult to explore on the sub-block level because every boundary is interdependent. But on the MB level, parallel methods are found as follows:

(a) The operation of each sub-block is identical so that they can be executed in parallel. But the correlation of data determines the order of filtering. In addition, memory access should also be taken into consideration.

(b) The vertical boundaries of vertically neighboring sub-blocks are uncorrelated, and the horizontal boundaries of horizontally adjacent sub-blocks are uncorrelated, too. For example, as shown in Fig. 4 (a), the left boundary of sub-block I-1 is uncorrelated to the left boundary of sub-block II-3, and the top boundary of sub-block I-4 is uncorrelated to the top boundary of sub-block III-7. So the parallelism can be used when filtering the vertical or horizontal boundaries.

(c) The steps of the filtering operation are the same for each boundary of the rows because data are uncorrelated between different rows. The same characteristic also exists in the filtering operation of columns.

(2) Optimization Methods of Deblocking

From the parallelism analysis of part (1), several different optimization methods can be considered:

(a) RP (Ranks Parallelism technology): Filtering of both vertical boundaries and horizontal boundaries can be pipelined. As shown in Fig. 4 (a), filtering of vertical boundaries (from a to d) is performed first, followed by that of horizontal boundaries (e to h).

(b) SBP (Sub-Block Parallelism technology): In the 4:2:0 video format, 48 edges must be filtered (including both luma and chroma). The filter order determines the efficiency and memory access. The order complies with what is shown in Fig. 4 (b) so as to maximize the utilization of memory.

(c) FGP (Fine-Grained Parallelism technology): Vertical edges can be filtered based on the unit of row at first; horizontal edges are filtered similarly in pipeline, as shown in Fig. 4 (c). Compared to methods (a) and (b), method (c) maximizes the utilization of parallelism of MB.

2.3 IDCT-IQ

(1) Parallelism Analysis of IDCT-IQ

IDCT-IQ has been used in many applications because of its simple architecture and low data-bit requirement. The whole operation (IDCT-IQ) can be divided into two parts: IQ first, then IDCT.

The integer DCT only refers to integer arithmetic, such as addition, subtraction, and shift, indicating low calculation strength. The basic unit of IDCT transform is 4×4 sub-block or 8×8. Taking IDCT 4×4 as an example, the calculation can be described in Eq. (1):

$$Y = CXC^T = \begin{Bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{Bmatrix} X \begin{Bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{Bmatrix} \quad (1)$$

The whole transforming can be realized in two steps: row transforming ($M = CX$), then column transforming ($Y = MC^T$). The IDCT of each row/column is the same, but independent from other rows/columns; furthermore, different sub-blocks can be disposed in parallel. These characteristics make it possible to improve the computing efficiency.

(2) Optimization Methods of IDCT_IQ

Based on the analysis of part (1), two methods can be used to realize parallelism in IDCT-IQ:

(a) Taking the transform of a 4×4 block as an example, the total operation can be divided into two parts: row transforming first, then column transforming. The same calculation steps are carried out among all rows, as shown in Fig. 5 (a), and among all columns, as shown in Fig. 5 (b).

(b) The sixteen 4×4 blocks in an MB can be disposed at the same time in parallel. So the total efficiency can be improved greatly.

Based on the aforementioned analysis, although some differences exist among these algorithms (MC, Deblocking, IQ-IDCT), there are still some common aspects:

1) Relative independency between complex judgment and intensive calculation

2) Intensive and regular calculation

3) Relative independency of data

These characteristics fit well in the parallel operation and pipeline technology.

3. Parallelization on REMUS

3.1 Experimental Environment

These typical algorithms (MC, Deblocking, IDCT-IQ) were implemented on REMUS. Figure 6 (a) shows the architecture of REMUS, including one host processor, two PEAs, an EnD (Entropy Decoder), and some assistant modules, such as interrupt controller, DMA (Direct Memory Access) controller, and AXI2AXI (Advanced Extensible Interface) bus bridge. Host processor, implemented by ARM1176JZ, is a typical embedded RISC to carry out control-intensive tasks. PEA is a powerful dynamic reconfigurable array, where several different algorithms can be mapped into PEAs concurrently and carried out independently to achieve high performance. EnD is a configurable stream decoder, which enables high performance on entropy decoding such as CAVLC and CABAC.

The PEA consists of 256 PEs (Processing Elements) organized as 4 PE8 × 8s, routers which can be restructured through the context interface and data-path. PE8 × 8 is the basic unit, consisting of four parts: DBI (Data Buffering Interface), FCI (Fast Configuring Interface), PEs and TRs (Temp Registers). Figure 6 (b) only shows one PE8 × 8 unit. The other three units have the same architecture as this one.

DBI is a flexible data exchange unit with asymmetric FIFOs, which can prepare data from off-chip and on-chip memories, as well as temp- data registers. FCI, controlled by host processor, DMA, and PEs, takes charge of control flow. Context information used to configure FCI supports different kinds of applications, which can be updated dynamically. Fast configuration techniques are adopted in FCI for data parallelization, including Context Pre-fetching, Context Prediction, and Partial Configuration. PE adopts the generic ALU architecture and serves as the basic calculation unit controlled by the context to realize different functions which are listed in Fig. 6 (c). Routers transmit data between different PE rows, point-to-point. TR, which has the same number with the PE output registers, can collect the processing results of PEs in the same row and feed the reserved data into the input of PEs in the same column. TR also plays an important role to combine four PE8×8s into one PEA16×16. Horizontal scaling uses the shared TR array technology. As shown in Fig. 6 (b), the left-up PE8 × 8 unit combines with the right-up one by sharing 4 columns of temp registers. Vertical expansion is simple, which uses a group of multiplexers (from the bottom router) to determine whether the PE's outputs are traveling circularly or straightly to the next unit. By adopting these two kinds of scaling approaches, REMUS processor can realize extension easily.

Flexibility of the PEA, as a primary characteristic, can be observed in three ways: First, the function of the PEA can be changed by dynamically switching contexts. Second, PEs can perform the calculation in collaboration with others or individually. Third, four sets of PE8 × 8 can be dynamically combined together according to the requirement of data calculation so as to achieve algorithm-level parallelism. For example, for some relatively complex algorithms (such as Deblocking and MC), one PE8×8 is not big enough to accommodate the calculation. Therefore, two or more PE8 × 8s can be combined together accordingly to carry out this complex algorithm.

3.2 Implementing Algorithms on REMUS

Among above algorithms (MC, Deblocking, IDCT-IQ)

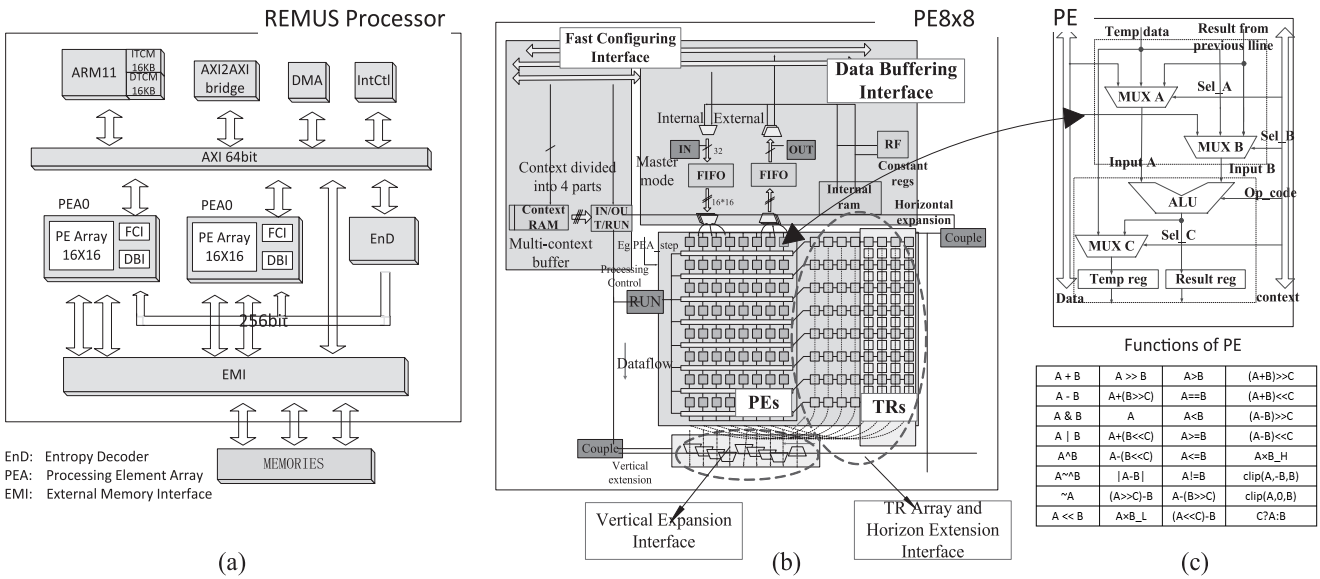


Fig. 6 (a) Architecture of REMUS, (b) PE array 8 × 8 simple diagram, (c) PE and PE's function.

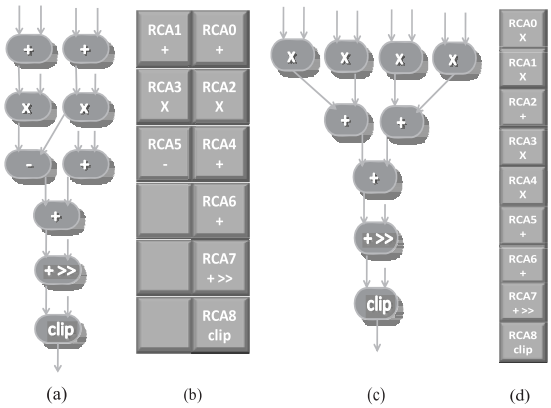


Fig. 7 (a) 6-tap filter DFG, (b) 6-tap filter mapping pic, (c) bilinear filter DFG, (d) bilinear filter mapping.

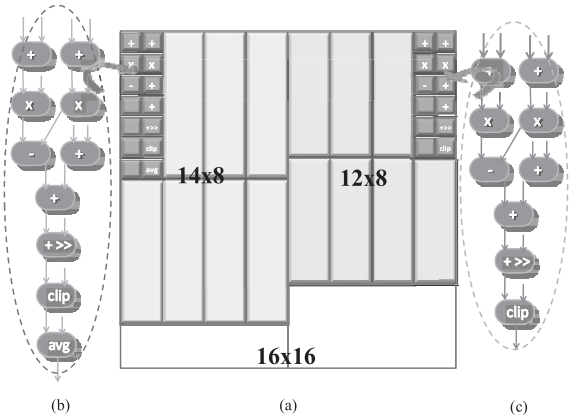


Fig. 8 (a) PEA16 × 16, (b) 6-tap filter + average, (c) 6-tap filter.

analysis, MC is the most typical algorithm because of its inherent characteristics, such as intensive calculation, high complexity and flexible control logic. Because of the limitation of space, only MC is taken as an example to illustrate the implementation procedures on REMUS according to the analysis in Sect. 2.

The first step is the HW/SW partition. The process of MC can be partitioned into two parts: computing the reference address and interpolation. The PEA is only suitable for the intensive and regular calculation, so the PEA can execute interpolation and the host processor can execute computation of the reference addresses.

The second step is to map interpolation (luma) onto PEA. In this process, there are several strategies: First, in order to make full use of pipeline technology, it is necessary to break down the calculation of the algorithm into relative independent steps and partition the PEA. As shown in Fig. 7 (a) and (b), the operation of the 6-tap filter can be divided into 6 stages. Each stage occupies one cycle. Second, it can take advantage of the sub-blocks parallelism because of the independence among sub-blocks. Take Fig. 3 (c) as an example: the operation of block0 and block1 is independent, so it can calculate the prediction values of these two blocks at the same time on the PEA. The PEA16 × 16s are divided into two parts: the left side (16 × 8) completes the calculation of block0, while the right side (16 × 8) completes block1. Figure 8 (b) and (c) shows calculations of block0 and block1. With the technology of pipeline, we can get the value in 15 cycles (block0) and 14 cycles (block1). Third, the calculation order of sub-blocks can be adjusted. For example, the original execution orders of Fig. 3 (c) is from 1 to 7, but the FPPs of block0 and block3 are the same, and the sequence can be arranged as follows: calculate the value of block1 by the left side (16 × 8) of the PEA at first, then block2 by dynamic configuration of the PEA. At the same time, on the right side (16 × 8) of the PEA, block0 and block3 can be calculated together. So the calculation value of MB, can be determined in 56 cycles (only calculation parts), adding the time of inner memory access (the time of input/output data), configuration and switch context of the PEA (12 cycles), the execution time is 96 cycles. It reduces 18 cycles compare to original order. Furthermore, the more complex of the same FPPs in different sub-blocks, the more time can be reduced. For example, if the FPP of block0 and block3 is (1/4, 2/4), changing the calculation order of sub-blocks can reduce 1/3 time.

The third step is to map the bilinear filter (chroma) onto PEA as follows: First, as with mapping luma on the PEA, the calculation can be reasonably divided into 5 relatively independent stages, as shown in Fig. 7 (c) and (d). Second, on the level of MB, for different chroma sub-blocks, the steps of calculation are similar. Making full use of PEA16 × 16, the PEA16 × 16 can be divided into 4 parts—four PE8 × 8s, with every part implementing the calculation of one sub-block. Furthermore, in order to achieve a higher efficiency, the 5 stages can be divided into 2 contexts: one context for the first 4 stages and the other context for the last stage. So the calculation values of the MB (chroma-Cb and chroma-Cr) can be determined in 12 cycles (calculation part).

To summarize, adding the time of inner memory access (the time of input/output data), calculation and switching context of the PEA (12 cycles), and utilizing the technology of parallelism and dynamic switch context text technology, in 4:2:0 format, the execution time of MB is 62 cycles at best (not the situation of examples), 168 cycles at typical, and 286 cycles at worst.

3.3 Implementing Results

Table 2 shows the MB execution results of the computing-intensive algorithms (MC, Deblocking and IDCT-IQ) on REMUS in the typical and worst case which are obtained by running decades of H.264 HiP 1080p's standard test sequences (note that all of the experimental cases are in YUV 4:2:0 formats). The dynamic reconfiguration technique is used during the implementation of these three algorithms on REMUS to reduce the execution time by making the parallelization of calculation and reconfiguration, for the reason that all of these algorithms are too complex to be disposed into one context (Note: since the dynamic reconfiguration

process and the computing process are overlapped, the duration of each dynamic reconfiguration is not listed in Table 2). Table 3 shows the comparison results of these three algorithms, on various architectures, which only contain the intensive calculation of the following algorithms: interpolation, boundaries filtering, and transform.

In Table 2, for MC, the host processor implements the irregular calculation of reference addresses, while the PEA performs regular and intensive interpolation. In the worst case, operation which contains complex calculation will consume 3×4 (luma) + 2 (chroma) = 14 sets of context and 6×4 (luma) + 2×2 (chroma) = 28 times of inner memory access. For Deblocking, the host processor executes the derivation of thresholds α and β, while the PEA executes the calculation of BS and the boundaries filter. The two sets of context correspond to BS = 4 and BS = 1, 2, 3 and one set corresponds to the boundaries filter. For IDCT-IQ, the host processor is in charge of data transform, and the PEA executes the integral transform. Considering the size of PEA16 × 16 and the independent characteristic of sub-blocks, 4 different 4×4 sub-blocks can be calculated in parallel by utilizing the technology of pipeline. The same is true for IDCT-IQ_8 × 8, except for some complexity. Furthermore, the execution time of both MC and Deblocking in the typical case is less than that in the worst case (about 13~40% reduction) for the reason that in the worst case, different complexities and situations, such as the I/P/B frames processing and MB partition, have to be taken into account, resulting in considerable computing work-load. But for the IDCT_IQ, the calculation steps are relatively fixed, therefore the execution time in the typical case is the same as that in the worst case.

Table 2 Results of complete algorithms.

	Host processor /cycles	Context /sets	Memory access /times	PEA /cycles
MC/MB	310	14	28	168(typical) 286(worst)
Deblocking /MB	305	3	6	387(typical) 445(worst)
IDCT_4X4 /MB	32	1	16	20 (typical/ worst)
IDCT_8X8 /MB	32	2	16	37 (typical/ worst)

In Table 3, for MC, the platform [10]-MC, ASIC [11]-MC, and XPP-PACT [4]-MC are all based on fixed block size. The efficiency of transmission will be limited when the partition is larger than this basic unit. By taking the method of variable block size according to MB partitions, the technology of parallelism (as mentioned in Sect. 2.1) and rapid calculation, the efficiency is improved by 60% compared with XPP-MC in the typical and worst case; compared with the ASIC_MC, the efficiency is improved by 70% in the typical case and 74% in the worst case; Compared with platform-MC, the efficiency is improved by 77.6% in the worst case. For Deblocking (intensive calculation part-bounder filter), XPP-Deblocking and ASIC-Deblocking [16] take traditional filtering steps, as mentioned in Sect. 2.2 methods(b)-SBP. On REMUS, FGP, the Sect. 2.2 methods(c)-FGP, is utilized. The efficiency is improved by 73% in the typical case and 69% in the worst case compared with XPP-PACT-Deblocking. For IDCT-IQ_4 × 4, XPP-PACT, Platform [10] -IDCT-IQ, and ASSP [17]-IDCT do not consider the parallelism among sub-blocks, but REMUS employs the method of parallel operation. The efficiency improved by 88.5% compared with XPP PACT, about 89.5% compared with Platform-IDCT-IQ, and about 17% compared with ASSP. From the frequency's perspective, REMUS can meet the requirements of H.264 HiP 1080p@30 fps at the 200 MHz working frequency by

Table 3 Comparison of different architecture results (cycle).

	PlatForm[10]	ASIC-MC[11]	ASIC-Deblock-ing[16]	ASSP[17]-IDCT	XPP-PACT[5]	REMUS
Architecture	ARM+IP cores	Hardware	Hardware	Hardware	Reconfigurable system	Hostprocessor +PEA
Working Frequency/MHz	130MHz	100MHz	200MHz	N/A	400MHz	200MHz
Interplator-MC	1280 (worst)	560(typical) 1120(worst)	N/A	N/A	424(typical) 720.8(worst)	168(typical) 286(worst)
Deblocking-Filter	N/A	N/A	192(worst)	N/A	688(typical/ worst)	184(typical) 212(worst)
IDCT-IQ_4x4	210	N/A	N/A	24(IDCT) (typical/ worst)	192(typical/ worst)	20(IDCT)+2(IQ) (typical/ worst)

making full use of parallel computing and efficient storage control mechanism. To reach the same performance, XPP has to adopt the 400 MHz working frequency and utilize a 120 MB reconfigurable buffer.

4. Conclusions

To achieve high performance parallel computing on reconfigurable system, it is very important to analyze the intrinsic parallelism of the algorithms, not only between host processor and PEA, but also among the different PEs. In order to realize the real-time decoding of H.264 HiP on REMUS, this paper examined the parallelization of MC, Deblocking, and IDCT-IQ, and proposed parallelization methods for these algorithms, such as variable size block partition, unfixed sub-block operation, sub-block-based parallelization for MC, MB-based parallelization for Deblocking and sub-blocks-based parallelization for IDCT. Furthermore, the proposed parallelization technology and HW/SW partition method could also be applied to MPEG2 and AVS (Audio Video coding Standard) [20] decoding. Tests showed that 1080p@30 fps of H.264 HiP@ Level 4, 1080p@30 fps of AVS Part-2 Jizhun Profile@ Level 4, and 1080p@30 fps of MPEG-2 MP@ High level decoding could be achieved on REMUS when exploiting a 200 MHz working frequency.

Acknowledgments

This work was supported by NNSF of China grant 60803018 and 863 Program of China grant 2009AA011702.

References

[1] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G.J. Sullivan, "Performance comparison of video coding standards using Lagrangian coder control," Proc. 2002 International Conference on Image Processing, 2002, vol.2, pp.II-501–II-504, 2002.

[2] Joint Video Team of ITU-T and ISO/IEC JTC 1, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, 2003.

[3] X.S. Zhou, E.Q. Li, and Y.K. Chen, "Implementation of h.264 decoder on general-purpose processors with media instructions," in Image and Video Communications and Processing 2003, Pts 1 and 2. vol.5022, ed. B. Vasudev, et al., Bellingham, Spie-Int Soc Optical Engineering, pp.224–235, 2003.

[4] "DaVinch," Texas Instruments, Feb. 2010.

[5] M.K.A. Ganesan, S. Singh, F. May, and J. Becker, "H. 264 decoder at HD resolution on a coarse grain dynamically reconfigurable architecture," International Conference on Field Programmable Logic and Applications, 2007, FPL 2007, pp.467–471, 2007.

[6] H. Singh, M.H. Lee, G.M. Liu, F.J. Kurdahi, and N. Bagherzadeh, "MorphoSys: An integrated reconfigurable system for data-parallel and computation-intensive applications," IEEE Trans. Comput., vol.49, no.5, pp.465–481, May 2000.

[7] M. Berekovic, A. Kanstein, D. Desmet, A. Bartic, B.F. Mei, and J.Y. Mingolet, "Mapping of video applications on a ADRES warse array," Workshop of Multimedia and Stream Processors, MSP7, Nov. 2005.

[8] Y.-H.C. Tzu-Der Chuang, C.-H. Tsai, Y.-J. Chen, and L.-G. Chen, "Algorithm and architecture design for intra prediction in H.264/AVC high profile," PCS2007, 2007.

[9] D.J. Zhou, Z.Y. You, J.Y. Zhu, J. Kong, Y. Hong, X.M. Chen, X.W. He, C. Xu, H. Zhang, J.J. Zhou, N. Deng, P.L. Liu, and S. Goto, A 1080p@60 fps Multi-Standard Video Decoder Chip Designed for Power and Cost Efficiency in a System Perspective, Japan Society Applied Physics, Tokyo, 2009.

[10] S.H. Wang, W.H. Peng, Y.W. He, G.Y. Lin, C.Y. Lin, S.C. Chang, C.N. Wang, and T.H. Chiang, "A platform-based MPEG-4 advanced video coding (AVC) decoder with block level pipelining," Proc. 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia, vol.1, pp.51–55, 2003.

[11] S.Z. Wang, T.A. Lin, T.M. Liu, and C.Y. Lee, "A new motion compensation design for H.264/AVC decoder," 2005 IEEE International Symposium on Circuits and Systems, pp.4558–4561, IEEE, New York, 2005.

[12] J.R. Hauser and J. Wawrzynek, "Garp: A MIPS processor with a reconfigurable coprocessor," Proc. 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines, pp.12–21, 1997.

[13] C. Ebeling, D.C. Cronquist, P. Franklin, J. Sewsky, and S.G. Berg, "Mapping applications to the RaPiD configurable architecture," Proc. 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines Cat. No.97TB100186, pp.106–115, Napa Valley, CA, 1997.

[14] T. Sato, et al., Implementation of dynamically reconfigurable processor DAPDNA-2, IEEE, New York, 2005.

[15] F. Vabid, G. Stitt, and R. Lysecky, "Warp processing: Dynamic translation of binaries to FPGA circuits," Computer, vol.41, pp.40–46, July 2008.

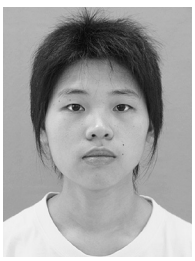
[16] G. Khurana, A.A. Kassim, T.P. Chua, and M.B. Mi, "A pipelined hardware implementation of in-loop deblocking filter in H.264/AVC," IEEE Trans. Consum. Electron., vol.52, no.2, pp.536–540, 2006.

[17] S.D. Kim, J.H. Lee, J.M. Yang, M.H. Sunwoo, and S.K. Oh, "Novel instructions and their hardware architecture for video signal processing," IEEE International Symposium on Circuits and Systems, ISCAS 2005, vol.4, pp.3323–3326, 2005.

[18] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.7, pp.560–576, 2003.

[19] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," IEEE Trans. Circuits Syst. Video Technol., vol.13, no.7, pp.614–619, 2003.

[20] Chinese Standard Press, "Information technology — advanced coding of audio and video — Part 2: Video," vol.GB/T 20090.2-2006, 2006.



Tongsheng Geng received the B.S. degree in Mechanical and Electrical Engineering from Zhengzhou University, Henan, China, in 2006, While she is currently working toward the M.S. degree in the Institute of Microelectronics, Tsinghua University, Beijing, China. Her research interests include reconfigurable computing and multimedia processing.

Leibo Liu received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree in Institute of Microelectronics, Tsinghua University, in 2004. He now serves as an Associate Professor in Institute of Microelectronics, Tsinghua University. His research interests include Reconfigurable Computing, Mobile Computing and VLSI DSP.

Shouyi Yin received the B.S., M.S. and Ph.D. degrees in Electronic Engineering from Tsinghua University, China, in 2000, 2002 and 2005 respectively. He has worked in Imperial College London as a research associate. Currently, he is with Institute of Microelectronics at Tsinghua University as an assistant professor. His research interests include mobile computing, wireless communications and SoC design. Dr. Yin has published more than 20 refereed papers, and served as TPC member or reviewers for the international key conferences and leading journals.

Min Zhu received the B.S. degree from the Department of Micro & Nano Electronic, Tsinghua University, Beijing, China, in 2006, where he is currently working toward the Ph.D. degree in the Institute of Microelectronics. His research interests include reconfigurable computing and multimedia processing.

Shaojun Wei was born in Beijing, China in 1958. He received Ph.D. degree from Faculte Polytechnique de Mons, Belguim, in 1991. He became a professor in Institute of Microelectronics of Tsinghua University in 1995. He is a senior member of Chinese Institute of Electronics (CIE). His main research interests include VLSI SoC design, EDA methodology, and communication ASIC design.