SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2014, Vol. 57 082406:1–082406:14
doi: 10.1007/s11432-013-4968-5

Implementation of multi-standard video decoder on a heterogeneous coarse-grained reconfigurable processor

LIU LeiBo[1], CHEN YingJie[1], WANG Dong[1]*, YIN ShouYi[1], WANG Xing[2], WANG Long[2], LEI Hao[3], CAO Peng[4] & WEI ShaoJun[1]

[1] *Institute of Microelectronics, Tsinghua University, Beijing* 100084*, China;*
[2] *Beijing Academy of Information Science and Technology, Beijing* 100878*, China;*
[3] *School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an* 710049*, China;*
[4] *National ASIC System Engineering Research Center, Southeast University, Nanjing* 210000*, China*

Received May 24, 2013; accepted July 15, 2013; published online October 30, 2013

Abstract This paper proposes a task-based hybrid parallel and hybrid pipeline (THPHP) scheme to implement multi-standard video algorithms, including MPEG-2, H.264, and audio video coding standard (AVS), on a heterogeneous coarse-grained reconfigurable processor, called the reconfigurable multimedia system (REMUS). The proposed schemes greatly improve decoding performance and satisfy the real-time requirements of various high-definition (HD) video decoding standards. In THPHP, we propose both a task-based hybrid parallel scheme, in which macro-block (MB)-level, block-level, and sub-block-level decoding tasks are parallelized to improve data processing throughput, and a hybrid pipeline scheme, in which slice-level, MB-level, block-level and sub-block-level computations are pipelined to improve efficiency. Computation-intensive tasks, such as motion compensation, intra prediction, inverse discrete cosine transform, reconstruction, and deblocking filter, are implemented on two reconfigurable processing units, which are the core computing engines of REMUS. Thanks to the proposed schemes, the implementations can achieve H.264 high profile (HP) 1920×1080@30 fps streams, AVS Jizhun profile (JP) 1920×1080@39 fps streams, and MPEG-2 main profile (MP) 1920×1080@41 fps streams when working at 200 MHz frequency. Compared with XPP-III (a commercial reconfigurable processor), when implementing H.264 HD decoding, the performance and energy efficiency on REMUS are improved by 1.81× and 14.3×, respectively.

Keywords multi-standard video decoder, coarse-grained reconfigurable multimedia system, computation-intensive tasks, parallelization and pipelining, HW/SW partitioning

---

Citation Liu L B, Chen Y J, Wang D, et al. Implementation of multi-standard video decoder on a heterogeneous coarse-grained reconfigurable processor. Sci China Inf Sci, 2014, 57: 082406(14), doi: 10.1007/s11432-013-4968-5

---

1 Introduction

Allied to the boom in the multimedia market and rapid upgrading of video coding standards, there is a potential market demand for a video decoder supporting multiple standards (e.g., MPEG-2 [1], H.264 [2], audio video coding standard (AVS) [3]). The video decoder needs to satisfy the requirements not only

---

*Corresponding author (email: doonny1212@gmail.com) of current standards, but also of various unknown standards in the future. As we know, the application specific integrated circuit (ASIC) solution has very high energy efficiency with high performance and low power consumption however with inferior flexibility. On the other hand, the general purpose processor (GPP) solution and programmable DSP solution have high flexibility, but suffer from relatively poor energy efficiency. Considering both flexibility and energy efficiency, it is difficult to catch up with the rapid evolution of the multimedia market by relying on the ASIC, GPP, and programmable DSP solutions, or even a combination of these. Also, with the development of information technology and the arrival of the low-carbon economic era, integrated circuits (ICs) and systems that target high performance and low power consumption have widely penetrated all aspects of the national economy, national defense construction, and everyday life [4]. Nowadays, reconfigurable systems, which can perform computations in a specially designed reconfigurable processing unit (RPU) to greatly enhance performance and reduce power while retaining much of the flexibility, are becoming an attractive topic in the field of computing architecture. In most video decoding algorithms, some of the computation-intensive tasks, such as motion compensation (MC), intra prediction (IP), inverse discrete cosine transform (IDCT), reconstruction (REC), and deblocking filter (DF), account for a large portion of the total computation intensity, e.g., roughly 50-80% in standard benchmark evaluations. Meanwhile, these tasks share similar features, such as MB-based operations and an intrinsic regular calculation mechanism. All of these features are suitable for reconfigurable computing, where functions are reconfigured onto RPUs at run-time to exploit pipeline and parallel schemes. Hence, it is meaningful to explore the approaches for implementing different video decoding algorithms on a reconfigurable system. Moreover, the derived approaches should contribute positively to the implementation of other computation-intensive applications on reconfigurable systems, such as communication baseband processing, computer vision processing, and encryption/decryption applications.

Many reconfigurable multimedia systems and video decoding implementations on these systems have been proposed in the literature. In [5], a configurable architecture was proposed to explore a hardware and software co-design technique for an H.264 decoder. Task-based MB level pipeline and MB-based parallel techniques were adopted. H.264 decoding of 352×288(i.e., CIF format) @19.72 fps was achieved with variable length decoding (VLD) implemented in software, while the other computation-intensive tasks were executed by hardware accelerators. In [5], reconfiguration could easily be implemented by adding or deleting accelerator modules, which is more flexible than ASIC and also more efficient than the GPP based solution. However, there configurable functionality is still not very flexible. ADRES, proposed in [6], is a coarse-grained reconfigurable architecture, combining a very long instruction word (VLIW) processor with coarse-grained reconfigurable matrix. In [7], a reconfigurable array was used to accelerate dataflow-like kernels, i.e., MC, inverse transform (IT), and DF in parallel, while the VLIW processor executed the non-kernel code by exploiting instruction-level parallelism. Even though most computation-intensive tasks were mapped onto the reconfigurable array, the pipeline operations incurred high overheads and degraded the mapped kernel performance. For example, MC of the luminance component was performed on a fixed 4×4 sub-block rather than on variable blocks. As a result, the performance of ADRES was relatively poor owing to its high pipelining overhead. In [8], an H.264 decoding algorithm was implemented on an XPP-III [9,10], which contains 40 ALU processing array elements (PAEs), 16 RAM PAEs, and 8 function PAEs. This implementation can decode H.264 streams with high performance, i.e., 1920×1080@24 fps streams at 450 MHz working frequency. However, since this work only focused on the H.264 video standard, it does not offer a comprehensive study of the task level scheduling and mapping methods for multi-standard video decoding algorithms on reconfigurable architectures.

The main contribution of this paper is the proposal of THPHP, a task-based hybrid parallel and hybrid pipeline scheme to implement multi-standard (MPEG-2, H.264, and AVS) video decoding on a heterogeneous coarse-grained reconfigurable processor. To reduce synchronization overhead and improve decoding performance, multiple level pipeline techniques are introduced. First, a slice level pipeline technique is adopted between entropy decoding (ED) and its subsequent tasks; second, an MB level pipeline technique is used both between inverse quantization (IQ) and its subsequent tasks, and between motion vector prediction (MVP)/intra prediction mode calculation (IPMC) and its subsequent tasks; and third, block level pipeline and sub-block level pipeline techniques are adopted in RPUs to improve the performance of computation-intensive tasks. Besides pipelining techniques, multi-level parallelism is also proposed to further improve the computation throughput. Since IQ and MVP/IPMC are mutually independent, an MB-based parallel technique is adopted between them. Similarly, for one MB in RPUs, the chrominance and luminance components are mutually independent. The block-based parallel technique is therefore adopted between chrominance IP and luminance IP, and between chrominance DF and luminance DF. For MC, the luminance component of one MB may be partitioned into various sub-blocks which remain independent, and thus, a sub-block-based parallel technique can be exploited to improve performance. The proposed parallel and pipelining techniques effectively increase hardware utilization and substantially reduce the power consumption under a low working frequency, while still maintaining a very high performance. Measurement results on fabricated chips show that compared with the XPP-III, the performance and energy efficiency on REMUS are improved by 1.81×and 14.3×, respectively, when implementing H.264 HD decoding.

The rest of this paper is organized as follows. Section 2 reviews the architecture of REMUS, while the hardware (HW) / software (SW) partitioning strategies for the three video decoding algorithms are introduced in Section 3. Section 4 presents task level scheduling and mapping methods for H.264, AVS, and MPEG-2. The scheduling approach for the system is given in Section 5. Section 6 presents an implementation and performance comparison, while Section 7 concludes the paper.

2 Review of the REMUS architecture

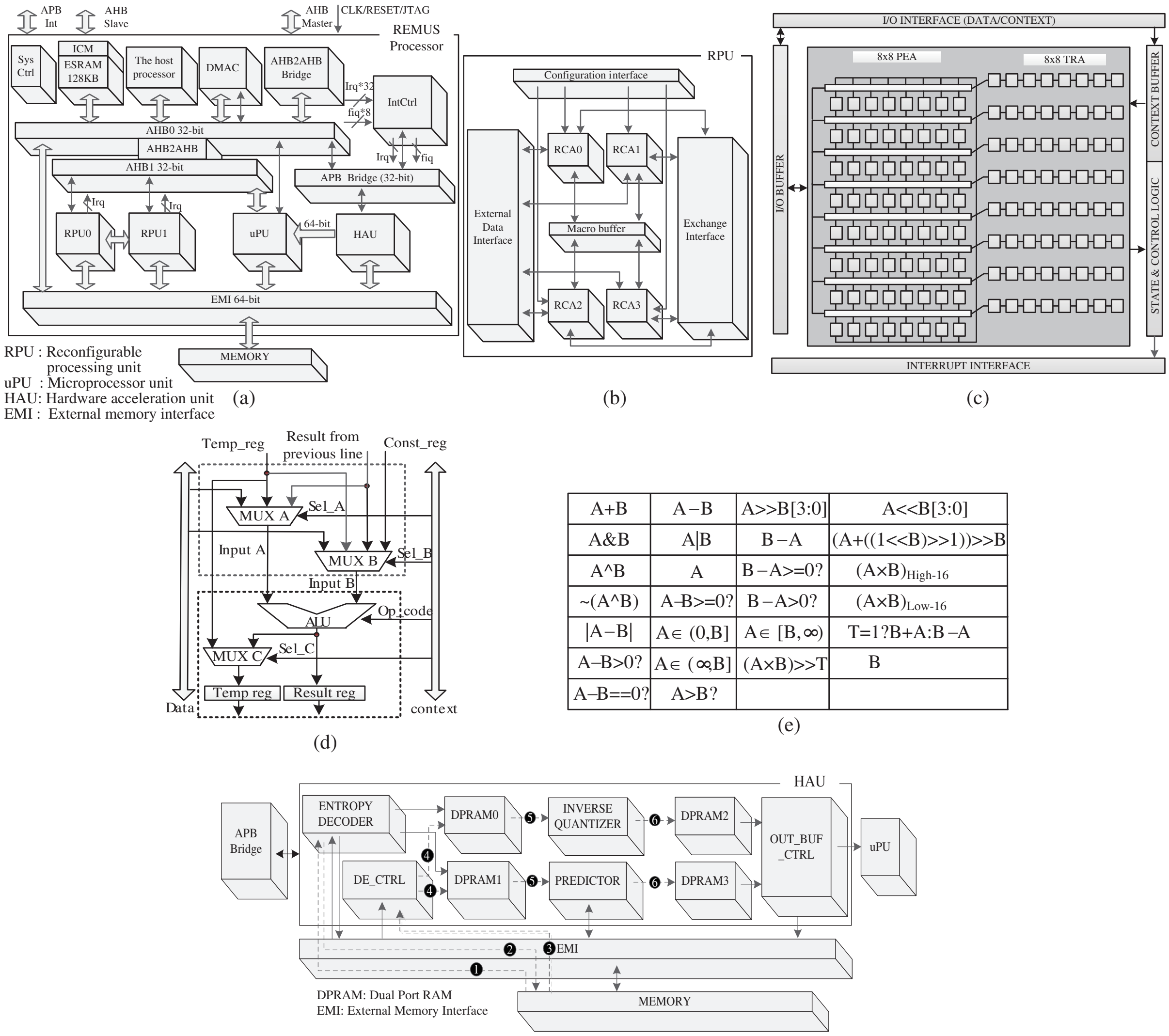
Figure 1(a) shows the architecture of REMUS. The REMUS processor is a coarse-grained dynamically reconfigurable processor [11], consisting of one host processor, two RPUs, a micro-processing unit (uPU), hardware acceleration unit (HAU), and other assistant modules, such as an interrupt controller (IntCtl), and a direct memory access controller (DMAC). These two RPUs are designed to implement computation-intensive and regular tasks. They can work in parallel or execute computation tasks sequentially. The HAU is a heterogeneous configurable module serving for control-intensive processes, while the host processor is responsible for simple operations, such as flow and interrupt controls.

2.1 Architecture of an RPU

The RPU is the core component for implementing computation-intensive decoding tasks. As illustrated in Figure 1(b), each RPU consists of four reconfigurable cell arrays (RCAs), a macro buffer, a configuration interface, an external data interface, and an exchange interface. The macro buffer is shared by four RCAs and used to exchange data among the RCAs. The configuration interface is used to load contexts, while the external data interface loads external data into an RPU or transfers data to the external memory. The exchange interface is the data exchange interface between the two RPUs. Each RCA consists of an 8×8 process element array (PEA), an 8×8 temp register array (TRA), an I/O buffer with 256-bit width, and some control modules as shown in Figure 1(c). The architecture of a processing element (PE) is illustrated in Figure 1(d), where each PE can realize 26 functions, as listed in Figure 1(e). One advantage of RCAs is to accelerate intensive parallel computations, especially loop calculations.

2.2 Architecture of the HAU

The HAU is mainly employed for control-intensive tasks, such as context-based adaptive variable length coding (CAVLC) and context-based adaptive binary arithmetic coding (CABAC), and IQ. Figure 1(f) shows the architecture of the HAU, including the entropy decoder, inverse quantizer, predictor, four DPRAMs with Ping-Pong mode, and so on. The entropy decoder can realize corresponding ED in H.264, AVS or MPEG-2, i.e., decoding of CAVLC, CABAC, context-based adaptive two-dimensional variable length coding (CA-2D-VLC), or Huffman. The inverse quantizer realizes inverse scanning (IS) and IQ in these three video standards, while the predictor implements MVP/IPMC and boundary strength calculation (BSC).



A+B	A−B	A>>B[3:0]	A<<B[3:0]
A&B	A|B	B−A	(A+((1<<B)>>1))>>B
A^B	A	B−A>=0?	$(A\times B)_{High\text{-}16}$
~(A^B)	A–B>=0?	B−A>0?	$(A\times B)_{Low\text{-}16}$
|A−B|	A∈ (0,B]	A∈ [B, ∞)	T=1?B+A:B−A
A–B>0?	A∈ (∞,B]	(A×B)>>T	B
A–B==0?	A>B?		

Figure 1 (a) Architecture of REMUS; (b) architecture of an RPU; (c) architecture of an RCA; (d) architecture of a PE; (e) PE functions; (f) architecture of the HAU.

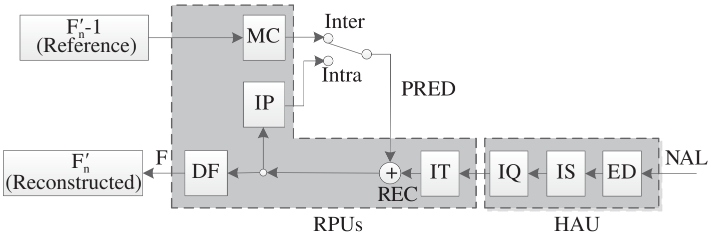


Figure 2 Block diagram of H.264 decoding process.

3 HW/SW partitioning strategies

Figure 2 shows a block diagram of the H.264 decoding process. The above MB-level information, i.e., sequence parameter set (SPS), picture parameter set (PPS), and slice headers, is parsed by the host processor because its symbol rates are very low [12]. ED parses and decodes the input bitstream to derive the syntax elements. IS performs an inverse scan of the residual samples of an MB. IQ implements

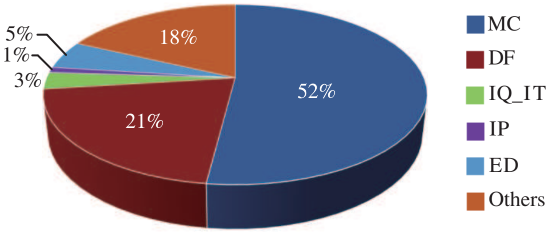


Figure 3 Time breakdown of H.264 reference decoder.

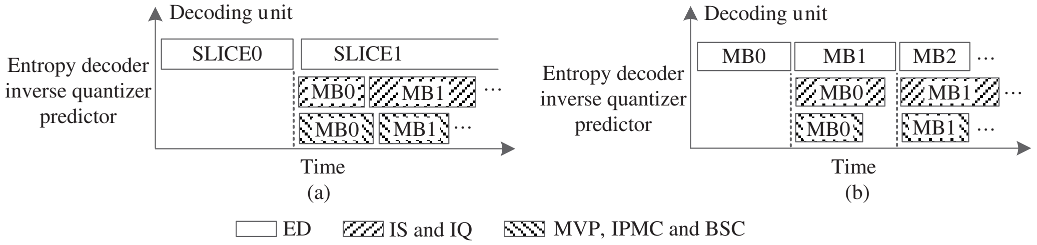


Figure 4 Task level scheduling on the HAU. (a) for H.264 or AVS video decoding; (b) for MPEG-2 video decoding.

inverse quantization of the residual samples after IS, and then, IT executes IDCT on the residual samples. MC and IP carry out inter- and intra-prediction, respectively, REC is used to reconstruct the MB, and DF implements a deblocking filter process to reduce blocking artifacts. Since the computation-intensive tasks (i.e., MC, DF, IQ, IT, and IP) account for approximately 77% of the execution time in H.264 HD decoding [13] as shown in Figure 3, these tasks are partitioned to be performed by two RPUs. The only exception is IQ. Since the required data-width of IQ exceeds the granularity of the RPU, it is executed by the HAU to avoid data overflow in the RPU. Regarding MC, only the interpolation calculations (intensive and regular calculations) are carried out in the RPU, while the reference address calculations (irregular calculations with many judgment branches) are performed in the uPU. Note that the MC referred to hereafter denotes the interpolation process. All the control-intensive tasks (i.e., ED, IS, MVP, IPMC, and BSC) with many conditional branches and irregular calculations are executed by the HAU. The HW/SW partitioning strategies for AVS and MPEG-2 are similar to those for H.264.

4 Task level scheduling and mapping methods for H.264, AVS, and MPEG-2 decoding

4.1 Task scheduling on the HAU

As described in Subsection 2.2, the HAU performs ED, IS, IQ, MVP, IPMC, and BSC in H.264, AVS, and MPEG-2 video decoding algorithms using the three function modules (i.e., entropy decoder, inverse quantizer, and predictor). From the perspective of these three video decoding algorithms, IS and IQ implemented by the inverse quantizer are independent of MVP/IPMC and BSC carried out by predictor. Therefore, an MB-based parallel technique can be used to implement the inverse quantizer and predictor. Task level scheduling on the HAU is illustrated in Figure 4.

In the H.264 or AVS video decoding process, a slice level pipeline technique is adopted between the entropy decoder and inverse quantizer/predictor to reduce synchronization overhead and improve the decoding performance, as shown in Figure 4(a). The black dotted line in Figure 1(f) shows the data flow between the entropy decoder and inverse quantizer/predictor:

Step1: The entropy decoder reads the original streams from memory by external memory interface (EMI) and decodes them with the outputs written into memory by EMI, as depicted by markers ① and ②.

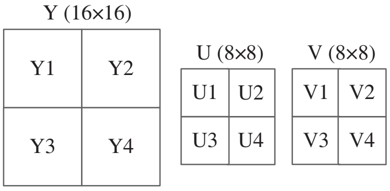


Figure 5 The block representation in one MB.

Step2: After the ED of slice 0 has been completed by the entropy decoder, DE_CTRL fetches data from memory by EMI, and then some data (i.e., the transform coefficients) are written into DPRAM0, while other information (e.g., MB type) is sent to DPRAM1, as demarcated by markers ③ and ④.

Step3: Markers ⑤ and ⑥ show the data flow of inverse quantizer/predictor, which fetches data from DPRAM0/DPRAM1 and then decodes the MBs in order with the outputs written to DPRAM2/DPRAM3.

In the MPEG-2 decoding process, an MB level pipeline technique is used between the entropy decoder and inverse quantizer/predictor as illustrated in Figure 4(b). In this case, the outputs of the entropy decoder are directly written into DPRAM0 and DPRAM1, rather than memory, which is different from H.264 or AVS decoding. And the data flow of the inverse quantizer/predictor in MPEG-2 decoding is the same as that in H.264 or AVS decoding. Note that there are no IP and DF processes in the MPEG-2 video decoding process.

4.2 Task scheduling and mapping on RPUs

4.2.1 *Task level scheduling and mapping methods on RPUs*

In this subsection, we make use of the notations shown in Figure 5, where Y refers to a 16×16 luminance block in one MB; Y1–Y4 denote luminance 8×8 blocks; U denotes the 8×8 chrominance U block; U1–U4 are the 4×4 chrominance U blocks; V is the 8×8 chrominance V block; and V1–V4 denote the 4×4 chrominance V blocks.

(1) H.264 decoding.

For H.264 decoding, RPU0 and RPU1 are configured in synchronous mode, i.e., MC/IP, IDCT and REC are implemented on RPU0, while the subsequent task DF is executed on RPU1, with the host processor controlling that fact that successive tasks are sequentially executed. Therefore, an MB level pipeline technique is used between RPU0 and RPU1. As discussed in Section 4.2.2, the actual decoding experiments show that the execution time of the decoding tasks on RPU0 and RPU1 are well balanced, which guarantees a very high pipelining efficiency.

Figure 6 shows the task level scheduling and mapping method for MC, IP, IDCT, and REC on RPU0. Six different mapping schemes are designed according to the corresponding MB type. The methods in Figure 6(a) and (b) are used for inter MBs, and those in (c)–(f) for intra MBs. For inter MBs, the four 8×8 luminance blocks are independent of each other when implementing MC, IDCT, or REC. This is also true for the four 4×4 chrominance blocks of U and the four 4×4 chrominance blocks of V. For an intra MB, if it is not an intra pulse code modulation (I_PCM) MB, the intra prediction mode for the luminance component may be intra_16×16, intra_8×8, or intra_4×4. Note that there are no IDCT, IP and REC processes if the MB type is I_PCM. In this case, the samples of the MB are transmitted, a mapping of which is shown in Figure 6(f). It can be seen from Figure 6(c)–(f) that RCA0 and RCA1 are in charge of decoding the luminance component of an intra MB; RCA2 takes charge of the chrominance component U of the MB; while RCA3 is responsible for the chrominance component V of the MB.

The above mapping method divides a task into several sub-tasks, which helps maximize utilization of the processing elements in the RPU and improves the performance by introducing block-based parallel, sub-block-based parallel, and block level pipeline and sub-block level pipeline techniques. For inter MBs, while executing MC, IDCT or REC, the utilization of the RPU can be up to 100%, that is, all of the four RCAs work in parallel. Note that an 8×8 luminance block can be further partitioned into different size sub-blocks, i.e., one 8×8 sub-block, two 8×4 sub-blocks, two 4×8 sub-blocks, or four 4×4 sub-blocks.

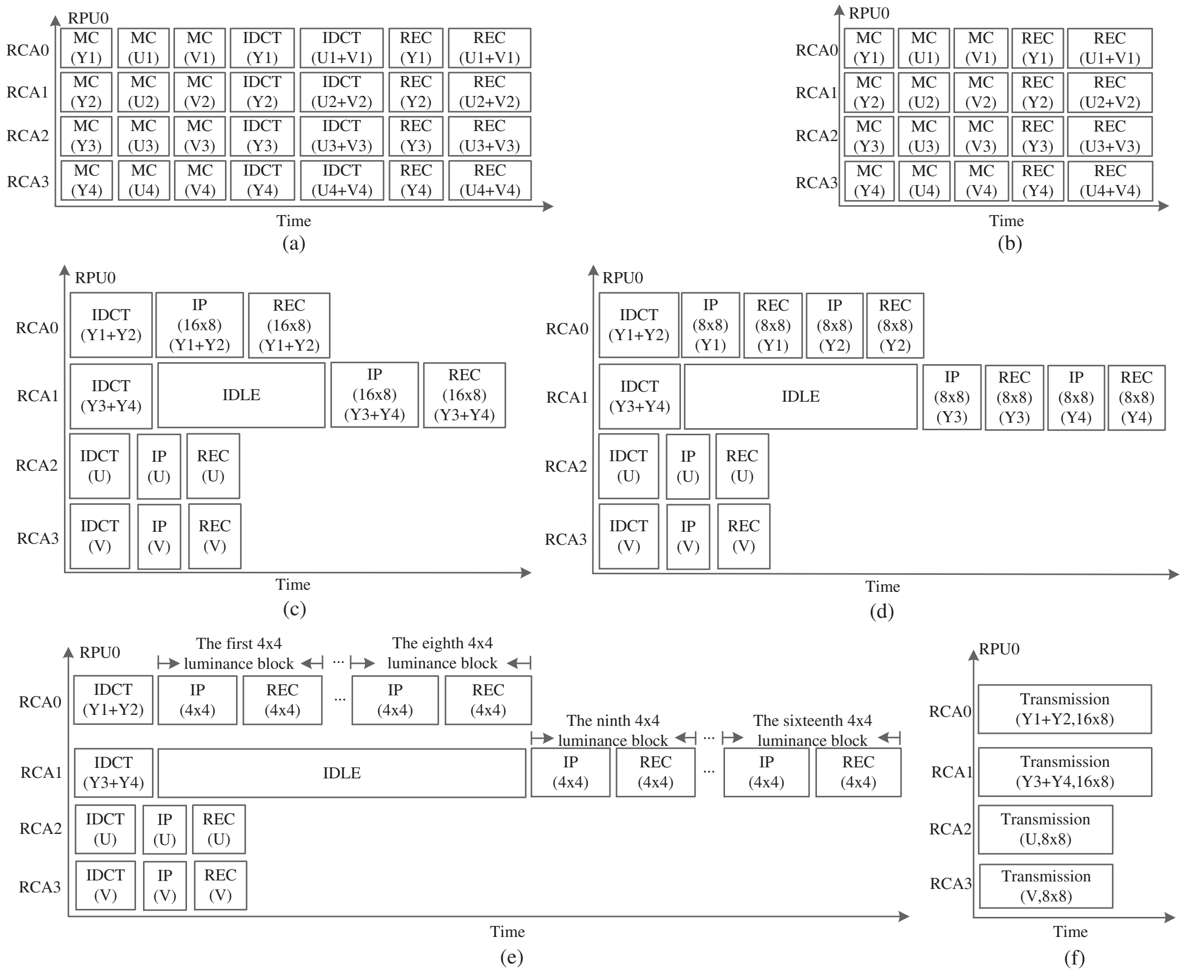


Figure 6 Task level scheduling and mapping method on RPU0. (a) Non-skipped inter MBs; (b) skipped MBs; (c) intra_16×16 MBs; (d) intra_8×8 MBs; (e) intra_4×4 MBs; (f) I_PCM MBs.

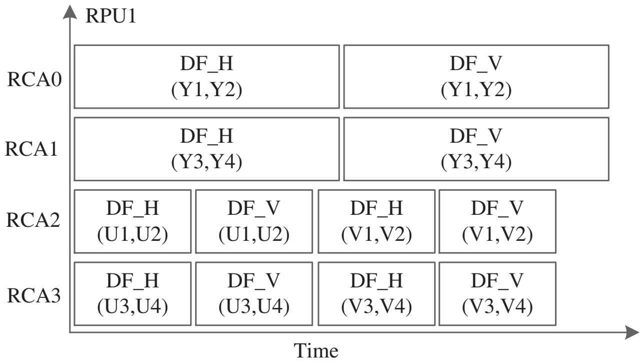
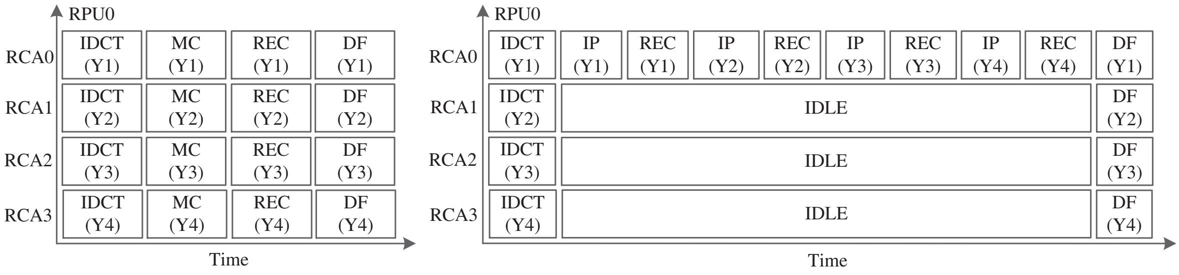


Figure 7 Task level scheduling and mapping method of DF on RPU1.

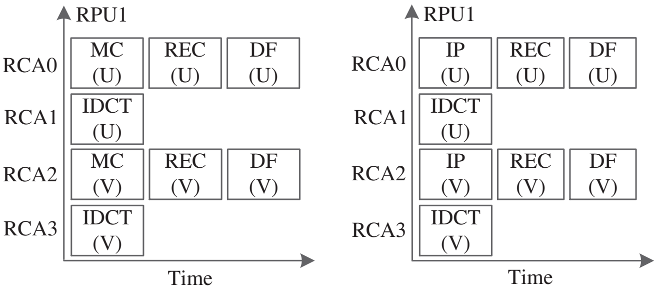
Corresponding to this, a 4×4 chrominance block is divided into one 4×4 sub-block, two 4×2 sub-block, two 2×4 sub-blocks, or four 2×2 sub-blocks. Hence, MC of an 8×8 size block in each RCA can be completed by carrying out the corresponding sub-tasks. In other words, the variable block size motion compensation (VBSMC) solution is adopted in MC to reduce the data bandwidth [5]. For intra MBs except for I_PCM, utilization of the RPU0 is 100% when executing IDCT, and decreases to 75% when executing IP and REC. For I_PCM MBs, the utilization of RPU0 is 100%.

Figure 7 shows the task level scheduling and mapping method of DF on RPU1. DF_H represents the



(a) (b)

Figure 8 Task level scheduling and mapping method on RPU0. (a) Inter MBs; (b) intra MBs.



(a) (b)

Figure 9 Task level scheduling and mapping method on RPU1. (a) Inter MBs; (b) intra MBs.

horizontal filter process for vertical edges, while DF_V denotes the vertical filter process for horizontal edges. It can be seen that DF is divided into several sub-tasks and that RPU1 utilization is 100%. Furthermore, block-based parallel and block level pipeline techniques are used to improve performance.

(2) AVS decoding.

During the AVS decoding process, RPU0 and RPU1 work in parallel synchronous mode. MC/IP, IDCT and REC of the luminance component are implemented on RPU0, while those of the chrominance components are carried out on RPU1. Therefore, a block-based parallel technique is adopted between RPU0 and RPU1.

Figure 8 shows the task level scheduling and mapping method for RPU0. The method in Figure 8(a) is applied to decode inter MBs. IDCT, MC, REC and DF of an 8×8 luminance block are executed on each RCA sequentially. The method in Figure 8(b) is the scheme used to decode intra MBs. In this case, IDCT and DF are each divided into four sub-tasks, which are executed on the four RCAs. While decoding inter MBs, the utilization of RPU0 is 100% when executing IDCT, MC, REC, or DF. While decoding intra MBs, the utilization of RPU0 is 25% when carrying out IP and REC and 100% for the IDCT or DF process. Block-based parallel and block level pipeline techniques are adopted in this mapping method.

As illustrated in Figure 9, in RPU1, RCA0 and RCA1 realize MC/IP, REC and DF of the chrominance component U, while RCA2 and RCA3 realize MC/IP, REC and DF of the chrominance component V. For chrominance components U and V, MC/IP and IDCT are implemented simultaneously to fully utilize RPU1. A task is divided into several sub-tasks; the utilization of RPU1 is 100% when implementing MC/IP together with IDCT, and 50% when carrying out REC or DF. Here block-based parallel and block level pipeline techniques are used to improve the performance.

(3) MPEG-2 decoding.

In MPEG-2 decoding, there are no IP and DF processes, thus it can meet the real-time decoding requirement when MC, IDCT and REC are executed on one RPU. During the MPEG-2 decoding process, RPU0 and RPU1 process different MBs, i.e., they work in parallel synchronous mode.

The task level scheduling and mapping method on RPU0/RPU1 is illustrated in Figure 10. It can be seen that a task is divided into several sub-tasks. During intra MB decoding, the utilization of RPU0/RPU1 is 100% when executing IDCT or REC of the luminance component. The utilization rate

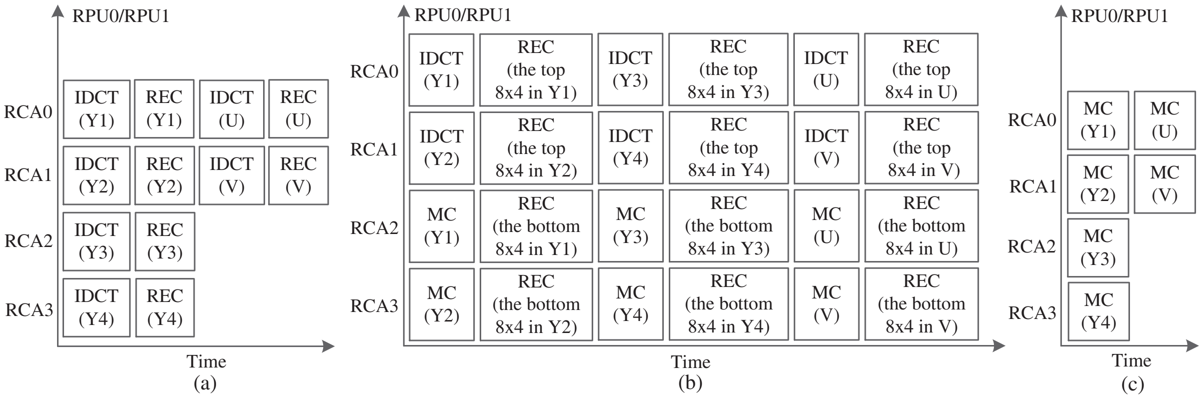


Figure 10 Task level scheduling and mapping method for RPU0 and RPU1. (a) Intra MBs; (b) non-skipped inter MBs; (c) skipped MBs.

Table 1 Number of cycles consumed by critical sub-algorithms in H.264 decoding

Sub-algorithm	Cycles/MB (with conventional approaches)	Cycles/MB (with proposed parallel and pipeline techniques)	Speedup
MC_4×4 (Sub-macro-block size is 4×4)	2968	1059	180%
MC_8×8 (Sub-macro-block size is 8×8)	628	323	94%
DF	1310	704	86%
IDCT4×4	144	106	36%
IDCT8×8	322	192	68%

decreases to 50% when implementing IDCT or REC of the two chrominance components. In non-skipped MB decoding, the utilization of RPU0/RPU1 is 100%. If decoding skipped MBs, the utilization of RPU0/RPU1 is100% and 50% when decoding the luminance and two chrominance components of a skipped MB respectively. Block-based parallel, sub-block-based parallel, block level pipeline and sub-block level pipeline techniques are adopted in this mapping method.

4.2.2 *Performance evaluation*

Table 1 shows the clock cycles consumed by critical sub-algorithms in H.264 decoding. It can be seen that the performance of the sub-algorithms is greatly improved by the proposed parallel and pipeline techniques. It should be noted that the MC, IDCT and DF (for H.264 and AVS only) sub-algorithms in MPEG-2 and AVS share similar data structures and computation patterns and could also be efficiently implemented using the proposed scheme. For brevity, we only report the measured performances for H.264 decoding since it is the most complicated and widely studied standard of the three.

5 System scheduling approach

The main process units in REMUS include the host processor, HAU, uPU, RPU0, and RPU1. The above MB-level information is parsed by the host processor. Control-intensive tasks (i.e., ED, IS, MVP/IPMC, and BSC) and IQ are executed by the HAU. Configuration contexts of MBs for the two RPUs are generated by the uPU. Computation-intensive tasks are carried out by RPU0 and RPU1.

The scheduling chart for the H.264 decoding process is illustrated in Figure11(a). A 4-stage MB level pipelining technique is introduced between the HAU, uPU, RPU0, and RPU1. DF of an MB is executed on RPU1. The other computation-intensive tasks (i.e., MC, IP, IDCT, and REC) of the MB are implemented on RPU0. Block-based parallel, sub-block-based parallel, block level pipeline and sub-block level pipeline techniques are used in RPU0 and RPU1 as mentioned in Section 4.

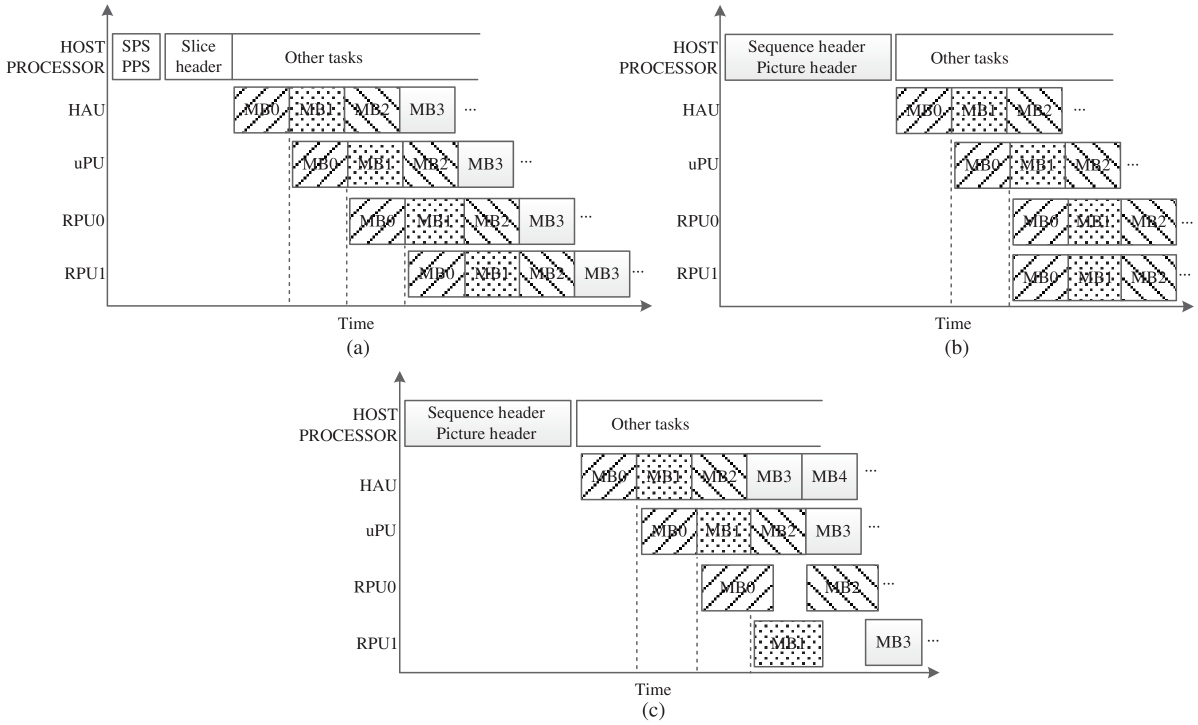


Figure 11 Scheduling charts. (a) For H.264 video decoding; (b) for AVS video decoding; (c) for MPEG-2 video decoding.

Figure 11(b) depicts the scheduling chart for the AVS decoding process. A 3-stage MB level pipelining scheme is used between the HAU, uPU and RPU0/RPU1, while an MB-based parallel technique is adopted between RPU0 and RPU1. In the AVS video decoding process, MC, IP, IDCT, REC, and DF of the luminance component of an MB are executed on RPU0, while those of the chrominance components of the MB are executed on RPU1.

Figure 11(c) shows the scheduling chart for the MPEG-2 video decoding process. A 3-stage MB level pipelining technique is used between HAU, uPU, and RPU0/RPU1. An MB-based parallel technique is applied between RPU0 and RPU1. All computation-tasks (i.e., MC, IDCT, and REC) are executed on an RPU, i.e., RPU0 and RPU1 process different MBs.

According to the discussion in Sections 4 and 5, the performance of multi-standard (i.e., H.264, AVS, and MPEG-2) video decoding can be increased by using the proposed scheme, which adopts the proposed slice level pipelining, MB level pipelining, block level pipelining, sub-block level pipelining, MB-based parallel, block-based parallel and sub-block-based parallel techniques correctly. All the techniques are based on tasks in the video decoding algorithms, which is the reason that the proposed scheme is called THPHP.

6 Implementation and comparison

Figure 12 shows a chip photograph of a multimedia targeted system on a chip, fabricated in TSMC 65 nm technology. Known as the **RHINOCEROS**, this chip is a reconfigurable high performance system and can be employed in a set-top box and communication base station, amongst others. It integrates an REMUS reconfigurable processor to support H.264, AVS, and MPEG-2 video decoding applications. **RHINOCEROS** also contains an ARM 11 processor core, PLL, and other circuits for peripheral controls as illustrated in Figure 12. The total area of REMUS is 48.9 mm^2. The areas of the host processor, HAU, uPU, and two RPUs occupy 0.2%, 6.0%, 15.3%, and 74.4% of REMUS, respectively.

6.1 Measured performance of the HAU

Measurements show that the decoding process of the HAU consumes 512, 435, and 438 clock cycles per

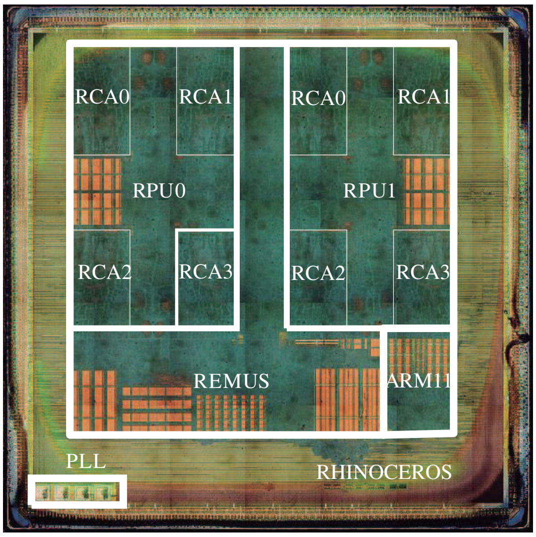


Figure 12 Chip photograph of **RHINOCEROS**.

Table 2 Throughput of CABAC running at 200 MHz

H.264 Stream	Resolution	File Size (bytes)	Frames	ABR (average bit rate, Mbps)	Total decoding time (ms)	Average decoding time per picture (ms)
Football	1920×1080	2 198 544	9	55.91	190.87	21.21
Fascination	1920×1080	3 437 008	15	52.44	325.46	21.70
Madagascar	1920×1080	1 400 308	8	40.06	81.83	10.23
Sun	1920×1080	10 297 103	30	78.56	817.88	27.26

Table 3 Sub-algorithms for H.264, AVS, MPEG-2 on an RPU

Sub-algorithm		H.264		AVS	MPEG-2
IDCT	Sub-macro-block size	4×4	8×8	8×8	8×8
	Cycles/MB	106	192	96	239
MC	Sub-macro-block size	4×4	8×8	8×8	8×8
	Cycles/MB	1059	323	204	191
DF	Cycles/MB	704		176	–

MB for H.264, AVS and MPEG-2 decoding respectively. The target decoding performance of H.264, AVS, and MPEG-2 is 1920×1080@30 fps when exploiting a 200 MHz working frequency, i.e., the time to process an MB should be limited to 816 cycles. The MB-level pipeline techniques are used in the decoding process, so the performance of the HAU satisfies the real-time decoding requirements of H.264, AVS, and MPEG-2. Of the three video algorithms, H.264 has the highest complexity, while the decoding performance of CABAC has a huge impact on the performance of the HAU in H.264 decoding. The bit rates of CABAC when decoding H.264 HD streams are listed in Table 2, with the average decoding time per picture listed in the last column. If the performance of H.264 video decoding is 30 fps, it means the average decoding time per picture is 33 ms. As the average decoding time per picture for these four streams is less than 33 ms, these four streams can be decoded and displayed in real-time.

6.2 Measured performance of critical sub-algorithms implemented on RPUs

Table 3 gives the performance of the critical sub-algorithms in H.264, AVS, and MPEG-2 video decoding, implemented on RPUs. The basic block size of IDCT is 8×8 in the AVS and MPEG-2 decoding processes,

Table 4 Sub-algorithms for H.264 on an RPU and XPP-III[9]

Sub-algorithm		RPU		XPP-III[9]		Speedup	
IDCT	Sub-macro-block size	4×4	8×8	N.A.		At least 8.85×	
	Cycles/MB	13+93	17+175	1700+192			
MC	Sub-macro-block size	4×4	8×8	4×4	8×8	4×4	8×8
	Cycles/MB	46+1013	42+281	2300+720	2000+364	1.85×	6.32×
DF	Cycles/MB	93+611		2400+688		3.39×	

while that in H.264 is 4×4 or 8×8. The different transform matrix and mapping methods of IDCT result in different performance in the H.264, AVS, and MPEG-2 decoding processes. The block size of MC in AVS is 8×8, that in MPEG-2 is 16×8 or 16×16, and that in H.264 may be 4×4, 4×8, 8×4, 8×8, 8×16, 16×8, or 16×16. If the block size is larger than 8×8, the block is divided into several 8×8 blocks, and the MC of these 8×8 blocks can be carried out in parallel. In the H.264 video algorithm, the 6-tap filter requires that (N+5)×(M+5) bytes reference data must be loaded for the interpolation of quarter-pixels in an N×M luminance block in the worst case. It is required that (8+5)×(8+5)=169 bytes reference data at the very least are loaded to predict an 8×8 luminance block. If the MC calculation of the 8×8 luminance block is based on a 4×4 block unit, it needs (4+5)×(4+5)×4=324 bytes reference data, which will increase the memory access cycles. Hence, the sub-macro-block size and the number of pixels processed per sub-macro-block have a huge impact on the performance of the MC sub-algorithm in the H.264 video algorithm, as seen in Table 3. The DF process is applied to all N×N block edges. In the H.264 decoding process, N is equal to 4 or 8 for the luminance component and 4 for the chrominance components. In the AVS decoding process, N is equal to 8. There is no DF process in the MPEG-2 decoding process.

6.3 Performance comparison

6.3.1 *Critical sub-algorithms in H.264 decoding*

The measured performance of the critical sub-algorithms in H.264 decoding is listed in Table 4. In the mapping process, the execution time consists of two parts. The first is latency, including the RPU reconfiguration time and pipeline filling time, and the second is calculation time. For example, the execution time of IDCT 4×4 on the RPU is (13+93) cycles, which means latency is equal to 13 cycles and calculation time is 93 cycles. The optimization schemes for configuration [14] to reduce latency are beyond the scope of this paper. The proposed optimization schemes for calculation time are described in Section 4. Compared with XPP-III, the performance of IDCT, MC and DF is improved at least 8.85×, at least 1.85×, and 3.39×, respectively.

6.3.2 *Decoding performance*

Table 5 gives a comparison between the XPP-based reconfigurable system in [8] and REMUS. The throughput of REMUS can achieve 30 fps when decoding H.264 HP 1920×1080 streams at a 200 MHz operating frequency. The unit MBs/s/MHz represents the video decoding capability, indicating how many MBs can be processed per second at the same clock frequency. This parameter is used to evaluate the decoding performance of a video decoder. Compared with [8], the H.264 decoding performance on REMUS is improved by a factor of 1.81. The unit MBs/s/mW (i.e., energy efficiency) denotes how many MBs can be processed per second with the same power consumption. The dynamic power (i.e., 3420 mW) of XPP-III in Table 5 is derived from 450 MHz × 7.6 mW/MHz [15]. Hence, the energy efficiency of REMUS is improved by 14.3× compared with that of XPP-III. When normalized to the same process (using a full-scaling approach), the energy efficiency of REMUS still outperforms that of XPP-III by about 4.76 times.

Table 5 Comparison of the performance of H.264 decoding using the system in [8] and REMUS

Item	System in [8]	Proposed
Platform	XPP-III	REMUS
Technology	90 nm	65 nm
Area(mm^2)	75	48.9
Clock	450 MHz	200 MHz
Performance	1920×1080@24 fps	1920×1080@30 fps
Power (mW)	3420	280
Normalized performance (MBs/s/MHz)	435	1224
Energy efficiency (MBs/s/mW)	57	874
Normalized performance comp.	1×	2.81×
Energy efficiency comp.	1×	15.3×

7 Conclusion

In this paper, we presented a scheme called THPHP to implement multi-standard (i.e., H.264, AVS, and MPEG-2) video decoding on REMUS, which is a coarse-grained dynamically reconfigurable multimedia processor. To accelerate the decoding process, several pipeline techniques, including slice level pipeline, MB level pipeline, block level pipeline and sub-block level pipeline techniques, were introduced. Moreover, several parallel techniques were applied to computation-intensive tasks to improve the decoding performance, including MB-based parallel, block-based parallel, and sub-block-based parallel techniques. Measurements on fabricated chips show that the proposed scheme supports H.264 HP 1920×1080@30 fps streams, AVS JP 1920×1080@39 fps streams, and MPEG2 MP 1920×1080@ 41fps streams in 4:2:0 format when exploiting an operating frequency of 200 MHz. The proposed techniques can also be exploited by other computation-intensive applications. Consider the upcoming HEVC standard for example, which is a natural extension of the H.264 standard and shares a similar decoding framework. According to our qualitative analysis, since both the operations and the calculation patterns are potentially supported by an RPU based reconfigurable fabric, motion compensation, adaptive interpolation, and transforms and quantization of HEVC (accounting for 70% of the total computation burden) can also be effectively mapped and realized using the proposed THPHP scheme.

Acknowledgements

This work was supported in part by China National High Technologies Research Program (Grant No. 2012AA012-701), Tsinghua Information S&T National Lab Creative Team Project, International S&T Cooperation Project of China (Grant No. 2012DFA11170), and NNSF of China (Grant No. 61106022).

References

1 ISO/IEC 13818. Generic Coding of Moving Pictures and Associated Audio (MPEG-2), 1994

2 Wiegand T, Sullivan G J, Bjontegaard G, et al. Overview of the H.264/AVC video coding standard. IEEE Trans Circ Syst Vid, 2003, 13: 560–576

3 Audio Video Coding Standard Workgroup of China. Information Technology-Advanced Coding of Audio and Video-Part2: Video, GB/T 200090.2–2006

4 Wang Y Y. The driving force for development of IC and system in future: reducing the power consumption and improving the ratio of performance to power consumption. Sci China Inf Sci, 2011, 54: 915–935

5 Jian G A, Chu J C, Huang T Y, et al. A system architecture exploration on the configurable HW/SW co-design for H.264 video decoder. In: Proceedings of IEEE International Symposium on Circuits and Systems, Taipei, 2009. 2237–2240

6 Mei B F, Vernalde S, Verkest D, et al. ADRES: an architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix. In: Proceedings of 13th International Conference on Field Programmable Logic and Applications, Lisbon, 2003. 61–70

7 Mei B F, Veredas F J, Masschelein B. Mapping an H.264/AVC decoder onto the ADRES reconfigurable architecture. In: Proceedings of International Conference on Field Programmable Logic and Applications, 2005. 622–625

8 Ganesan M K A, Singh S, May F, et al. H.264 Decoder at HD resolution on a coarse grain dynamically reconfigurable architecture. In: Proceedings of 2007 International Conference on Field Programmable Logic and Applications, Amsterdam, 2007. 467–471

9 PACT Company. XPP Technologies: White Paper of Video Decoding on XPP-III, 2006

10 Rossi D, Campi F, Spolzino S, et al. A heterogeneous digital signal processor for dynamically reconfigurable computing. IEEE J Solid State Circ, 2010, 45: 1615–1626

11 Zhu M, Liu L B, Yin S Y, et al. A cycle-accurate simulator for a reconfigurable multi-media system. IEICE Trans Inf Syst, 2010, E93–D: 3202–3210

12 Chen T W, Huang Y W, Chen T C, et al. Architecture design of H.264/AVC decoder with hybrid task pipelining for high definition videos. In: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, 2005. 2931–2934

13 Baik H, Sihn K H, Kim Y, et al. Analysis and parallelization of H.264 decoder on cell broadband engine architecture. In: Proceedings of IEEE International Symposium on Signal Processing and Information Technology, Giza, 2007. 791–795

14 Wang Y S, Liu L B, Yin S Y, et al. Hierarchical representation of on-chip context to reduce reconfiguration time and implementation area for coarse-grained reconfigurable architecture. Sci China Inf Sci, 2013, 56: 112401(20)

15 Schuler E. NoC concepts with XPP-III. In: Proceedings of International Symposium on Reliability of Optoelectronics for Space (ISROS 2009), Cagliari, 2009