Liu et al. EURASIP Journal on Image and Video Processing (2017) 2017:24
DOI 10.1186/s13640-017-0174-5

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

A novel architecture for parallel multi-view HEVC decoder on mobile device

Wei Liu, Jiao Li and Yong B. Cho*

Abstract

The multi-view HEVC (MV-HEVC) extension was finalized in July of 2014 by the Moving Picture Experts Group and the Video Coding Experts Group. Recently, multi-view videos based on stereo representations are becoming widely popular. Also, a variety of multimedia contents are now available for mobile devices. A real-time multi-view video decoder is therefore needed. In mobile devices, a real-time decoding multi-view video is difficult because of the increasing number of views, spatial resolutions, and limited speed of the processors on mobile platforms. In this paper, we propose a novel architecture for a real-time decoder in mobile devices. The proposed MV-HEVC decoder uses parallel-optimized multi-view video decoding with multi-threading, using advanced reduced instruction set computer machine (ARM) Cortex multi-core processors. Moreover, it is optimized in single instruction multiple data for an ARM platform. The proposed multi-core decoding architectures enable multi-threading with minimum processing overhead. Experimental results show that the proposed multi-view video coding increased the speed by around 2.4–4.8 times in the ARM platform compared to MV-HEVC.

Keywords: MV-HEVC, Multi-threaded, Multi-core, Real-time, SIMD, Mobile, ARM NEON

1 Introduction

With the rapid development of the mobile Internet, mobile platforms for video technologies are becoming increasingly popular. For this reason, the resolution of the mobile platform screen has been improving. 720p and 1080p screens have become the mainstream screen resolutions. The High Efficiency Video Coding (HEVC) compression scheme can improve the compression efficiency of 1080P video content by about 50% compared with H.264 [1]. HEVC is a video compression standard and is considered the successor of the H.264/MPEG-4, Advanced Video Coding International Telecommunication Union Telecommunication Standardization Sector (AVC ITU-T) standard. The first edition of the HEVC video compression standard in April 13, 2013, was accepted by the International Telecommunication Union (ITU-T), the official standard [2–4]. Figure 1 depicts a hybrid video encoder that can create a block diagram that complies with the HEVC standard data stream. Intraprediction involving the spatial domain pixel correlation of an image generally uses adjacent pixel decoding block elements as reference data and obtains the pixels of the current block unit value by interpolating. Although the HEVC intraprediction is similar to H.264, HEVC can predict up to 33 directional modes. Interframe prediction includes motion estimation and motion compensation. Although the motion estimation principle is the same as that of the H.264 standard, HEVC includes advanced motion vector prediction (AMVP) and merge mode [2]. AMVP uses data from the reference picture and can also use data from adjacent prediction blocks. The merge mode allows for the motion vectors (MVs) to be inherited from neighboring prediction blocks. The motion vector (Vector-MV Motion) in interframe prediction is produced by motion estimation, which is used to represent the offset of the corresponding block regions of the current prediction unit (PU) relative to the reference image. HEVC have the combined coding mode and improves the compression ratio with the same MV merging in the time domain. Unlike H.264, HEVC also contains improved skip mode and direct mode [2]. In terms of entropy coding, HEVC abandoned the H.264 context-based adaptive-variable length coding (CAVLC) but retained the more efficient Context-based Adaptive-

* Correspondence: ybcho@konkuk.ac.kr
Department of Electronics, Information and Communication Engineering, Konkuk University, 120 Neungdong-ro, Gwangjin-gu, Seoul 05029, South Korea

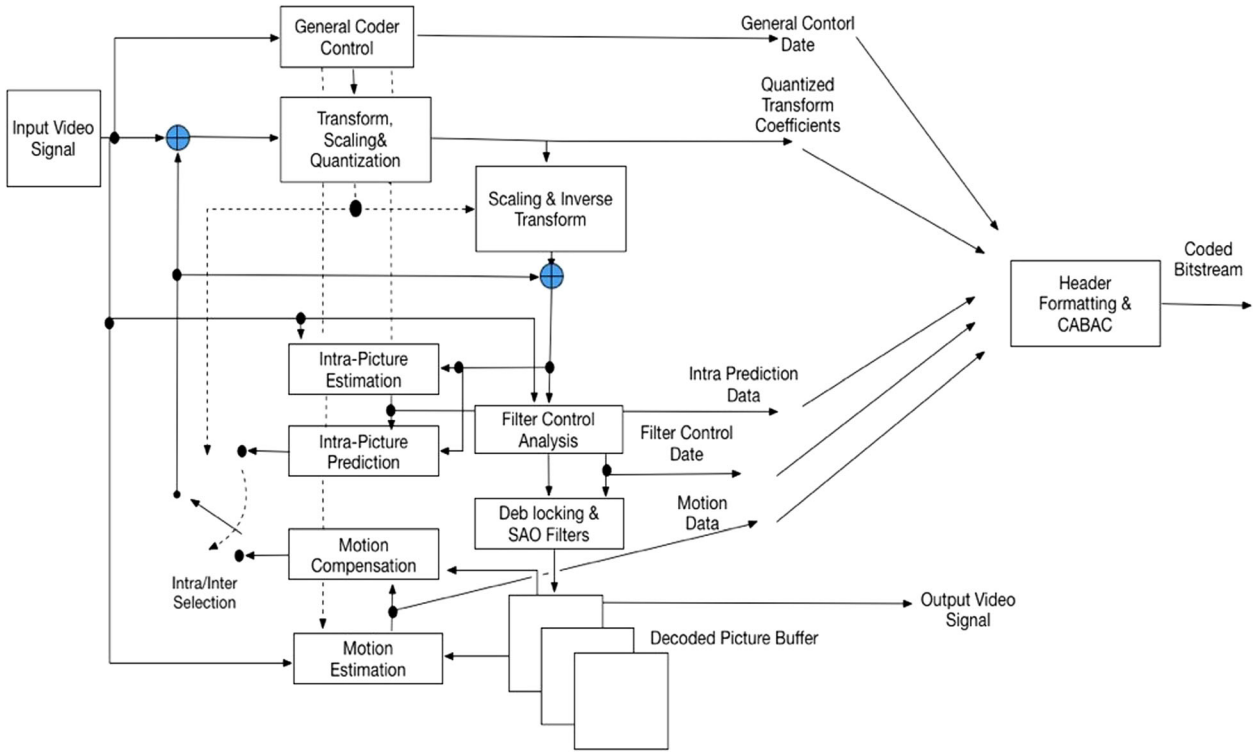

Fig. 1 Hybrid video encoder for HEVC. In the Section 1

binary Arithmetic Coding (CABAC) method, in addition to optimizing the speed and compression rate as well as the context of the storage [5]. With regard to the deblocking filter, HEVC and H.264 are similar, but HEVC uses an 8×8 sampling grid, which is more suitable for parallel operation [6]. Parallel processing architecture is one of the advantages of HEVC. HEVC provides three parallel processing tools: Slice, Tile, and WPP, which are conducive to the realization of coding development from single core to parallel multi-core. HEVC has become the mainstream of video coding. The second approved version of the HEVC/H.265 standard contains a multi-view extension profile [7], making a video multi-view possible on mobile terminals. In multi-view, a video can be watched from multiple angles. That is, the audience is no longer subject to a camera-specific point of view, and it is now possible to watch a video from any preferred angle. Compared with the traditional single-channel video, the multi-view video has a wealth of information in the form of stereo imaging and incorporates freedom of video data from the perspective of the scene as shown in Fig. 2. The Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) was established to work on multi-view and 3D video coding extensions for HEVC and other video coding standards. The multi-view extension of HEVC (MV-HEVC) enables encoding/decoding of 3D and multi-view video [4, 8]. However, the MV-HEVC (HTM 15.0), which is coded for x86/x64 Linux/Windows, is cross compiled for ARM Linux OS. Given a resolution of

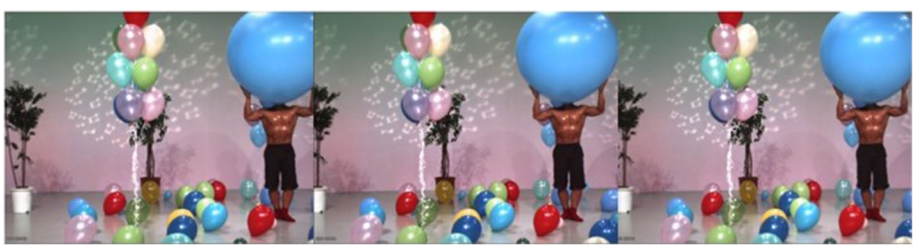

Fig. 2 Multi-view video sequence of balloon. In the Section 1

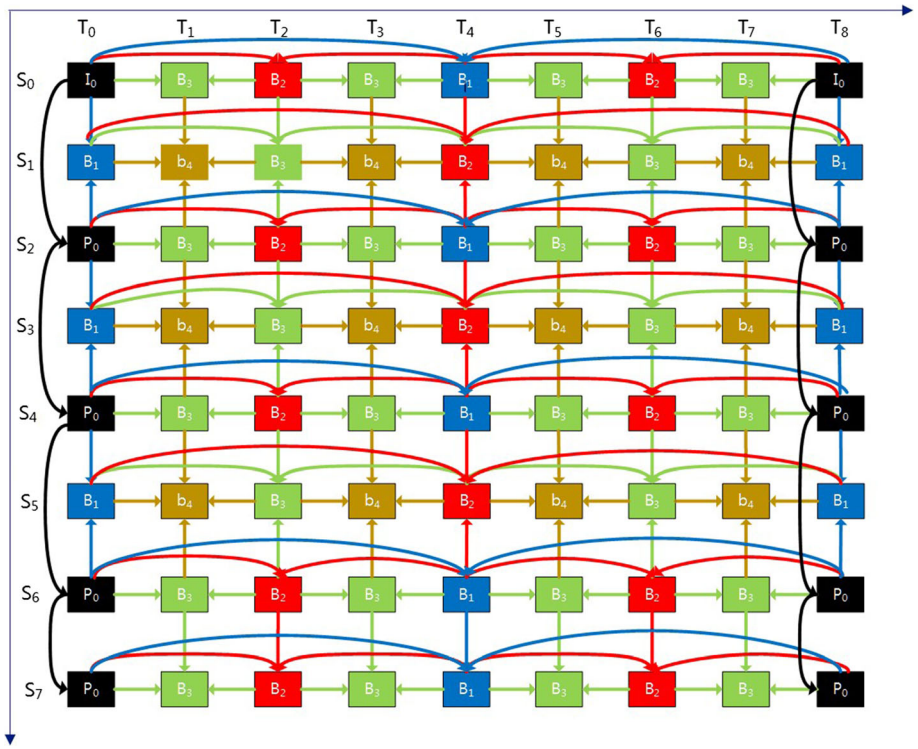

Fig. 3 Example of multi-view prediction structure. View S0 represents the basic views and pictures in a non-basic view that can be predicted from a dependent (base view) picture at the same time. Pictures denoted by "*I*" use only intra-picture prediction, pictures denoted by "*P*" refer to uni-predictive inter-picture prediction, and pictures denoted by "*B1*", "*B2*", "*B3*", and "*b*" refer to bi-predictive inter-picture prediction [4]. In the Section 1.1

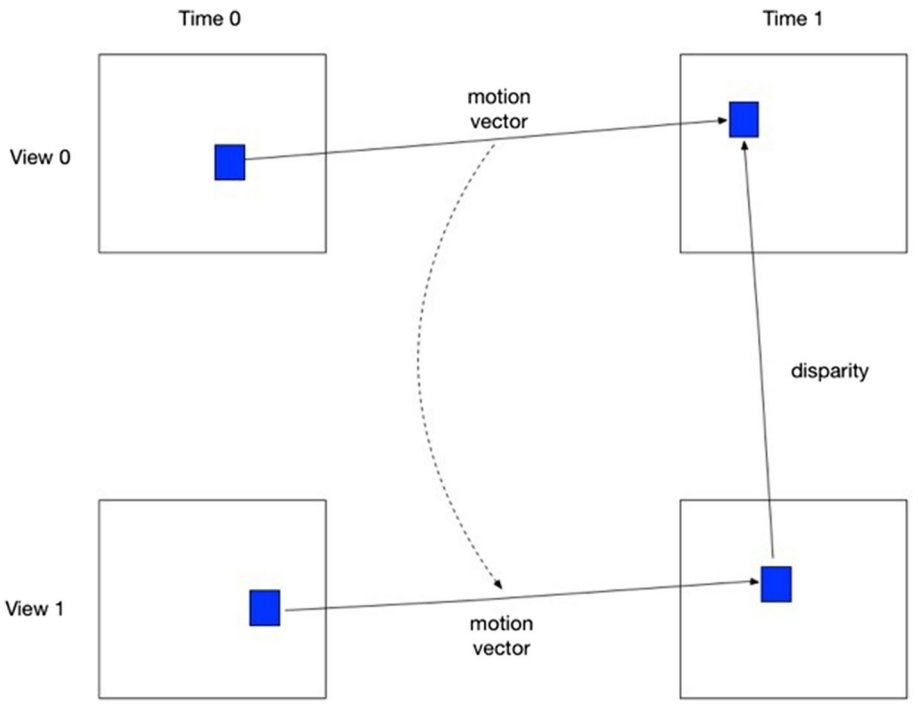

Fig. 4 Illustration of motion prediction between views, where the motion vector of view 1 is inferred from the motion vector of view 0 using corresponding blocks at time 1 based on the NBDV disparity between those blocks. In the Section 1.1

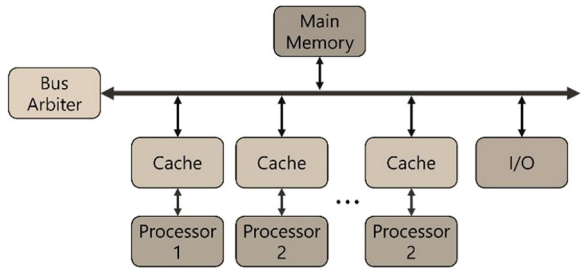

Fig. 5 Diagram of a symmetric multiprocessing system. In the Section 2

1024×768, the size can become too large for the ARM platform. With such a resolution, the decoding speed cannot meet the requirements of real-time applications. In this paper, we propose and demonstrate the use of multi-core processing to improve the decoding rate. We implement multi-core processing with multi-threading, which uses pthread (POSIX Threads) [9]. pthread is the POSIX standard. The standard defines a set of APIs for creating and manipulating threads. In the UNIX operating system (Linux, Mac OS X, UNIX, etc.), pthread is used as the operating system thread. When using multiple threads to implement multi-view video coding, it was found that MV-HEVC benefits from inter-view predictions but it is technically difficult to implement multi-view video coding. Another option is to use simulcast, which has no inter-view predictions but can be implemented with multi-threading. However, it has a lower bitrate than MV-HEVC [10]. Therefore, it is inefficient for compressing multi-view video sequences. In this paper, we redesign the codec method for multi-threading with ARM. Using multi-threading, the results of the decode rates have been improved by almost four times compared to the quad-core with single core when decoding an 8-view multi-view video. ARM NEON is an extended structure for a 128-bit single instruction multiple data (SIMD) (instruction multiple, single data, single instruction, multiple data) architecture. ARM Ubuntu from version 9.04 supports ARM NEON [11, 12]. SIMD is a single instruction multiple data stream, capable of copying multiple operands and packing them into a set of instruction sets in a large register. When we use a SIMD type of CPU, the instruction decoding process for several parts of the memory involves one-time access to all the operations. This feature makes SIMD particularly suitable for data-intensive operations such as multimedia

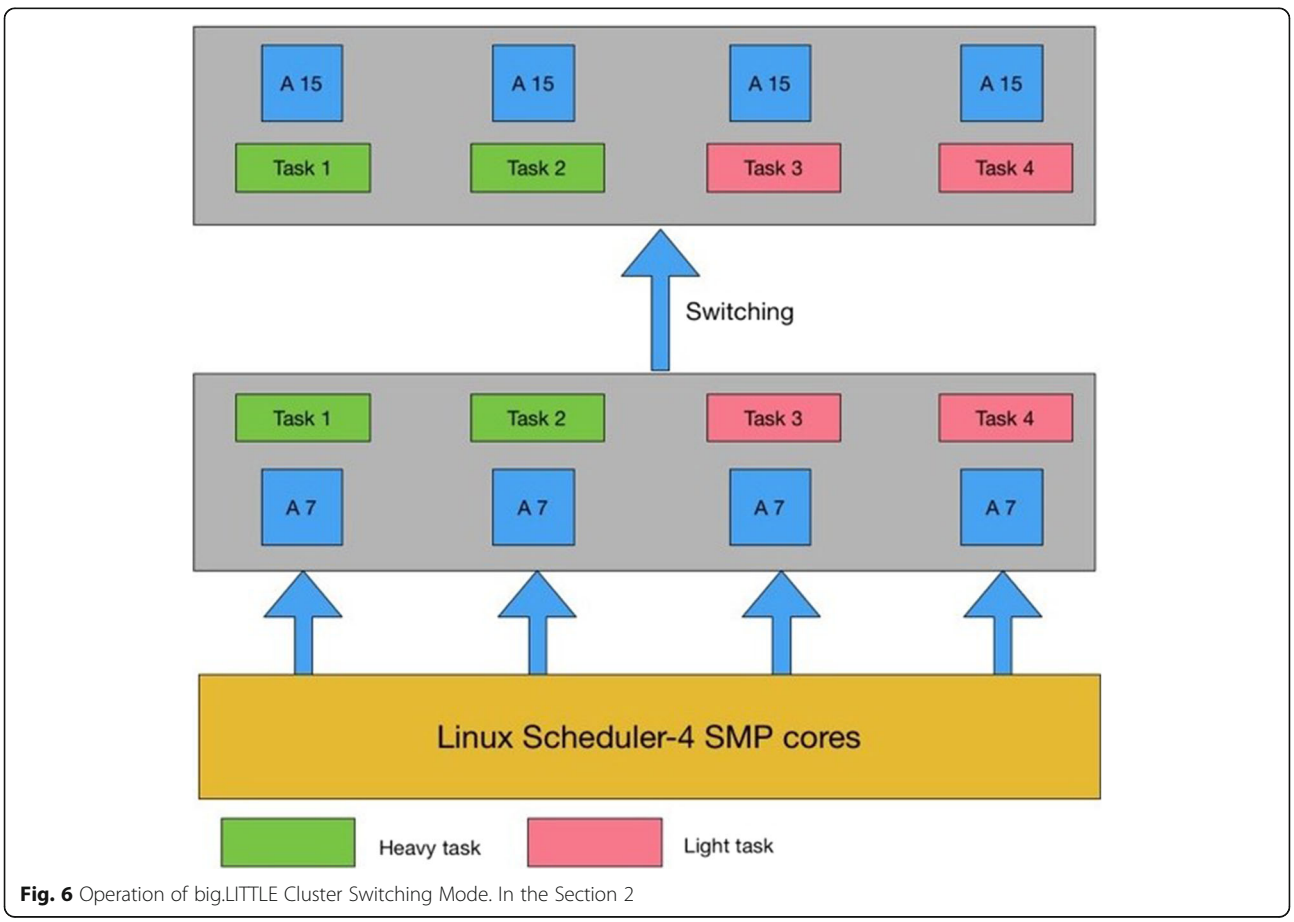

Fig. 6 Operation of big.LITTLE Cluster Switching Mode. In the Section 2

Table 1 Comparison between cluster switching mode and HMP mode

	Cluster switching	HMP mode
Configuration	One cluster is activated at a time	All cores work independently
Kernel impact	Minimum modification in Linux kernel	Linux Scheduler and CPU driver should be modified
Max performance	Sum of performance of all the big cores	Sum of performance of all the big and LITTLE cores
Switching/Migration	Switch by CPU frequency framework	Migrated by scheduler

applications [13]. In this paper, we show how to build a codec model for multi-threading. We also compare different performances under different CPU platforms. We further test the use of the SIMD structure to improve the speed of the specific situation. Our platform comparison is based on quad-core ARM Cortex-A15 and ARM dual core Cortex-A9. We also use ARM Ubuntu OS to test MV-HEVC sequences in 8, 6, and 4 views. The paper is organized as follows. Section 2 presents implementation environment with multi-core and new methods for MV-HEVC. Section 3 presents data-level parallelisms for MV-HEVC, and Section 4 shows the performance and numerical analysis of the proposed methods. Finally, Section 5 concludes the whole work.

1.1 Overview MV-HEVC

Multi-view video coding can allow users to freely choose to watch a video from different views. The simplest structure for a multi-view video is multi-view-HEVC (MV-HEVC), which is an HEVC extension. MV-HEVC uses the same design principles as multi-view video coding (MVC), which is an extension of AVC [14, 15]. With the MV-HEVC design, MV-HEVC adopts the reference picture lists (RPL) construction to modify the inter-view prediction, such that pictures from other views at the same time instances can be used for prediction. This means that any disparity shifts between the views is compensated for in the prediction process [4]. The MVC has two types of inter-view prediction approaches [16, 17]. The first inter-view prediction model predicts the structure as shown in Fig. 1. In this model, the inter-view prediction is enabled by HEVC's flexible reference image management function. During prediction processing, the other view from the decoded picture is inserted into the reference picture, which lists the construction of the current view as shown in Fig. 3. Therefore, the reference picture that lists the construction that includes the current view for the temporal reference pictures can be used to predict the current picture while showing neighboring views simultaneously. With block-level coding modules, it is possible to use the correlation between the view motion and residual data. Scene objects projected in different views have similar characteristics of motion and texture as shown in Fig. 4. Therefore, the

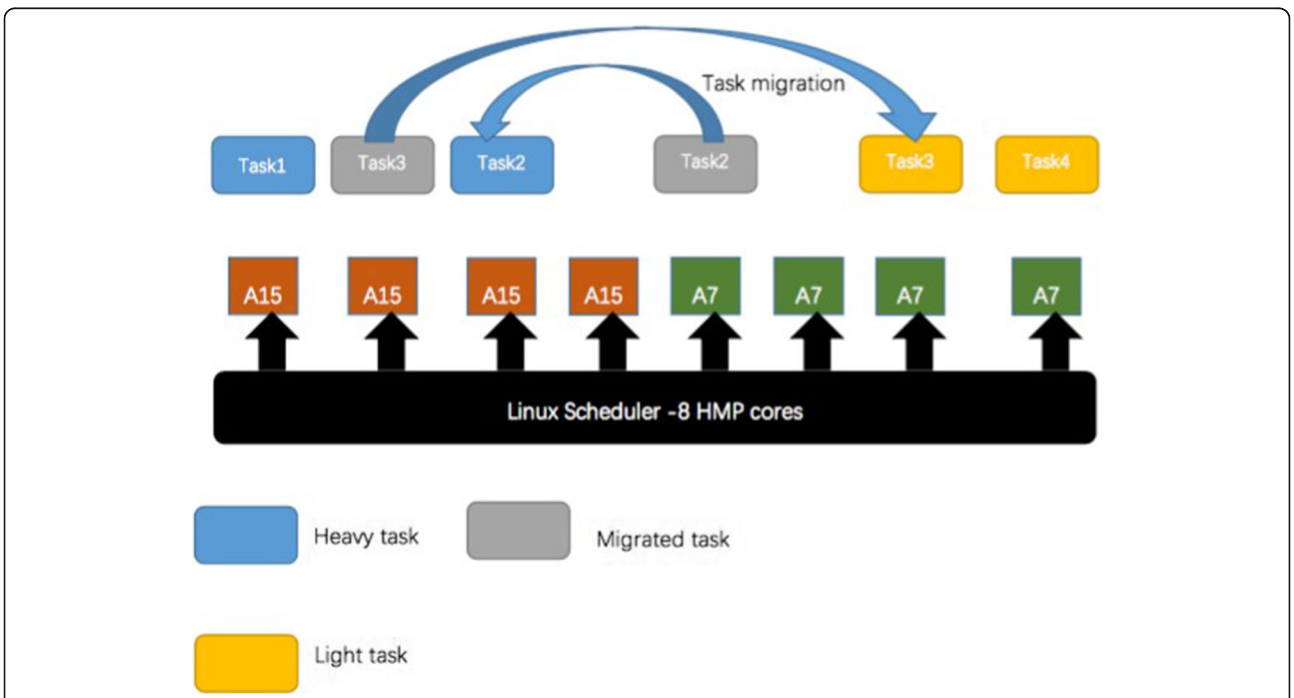

Fig. 7 Operation of BIG/LITTLE HMP Mode. In the Section 2

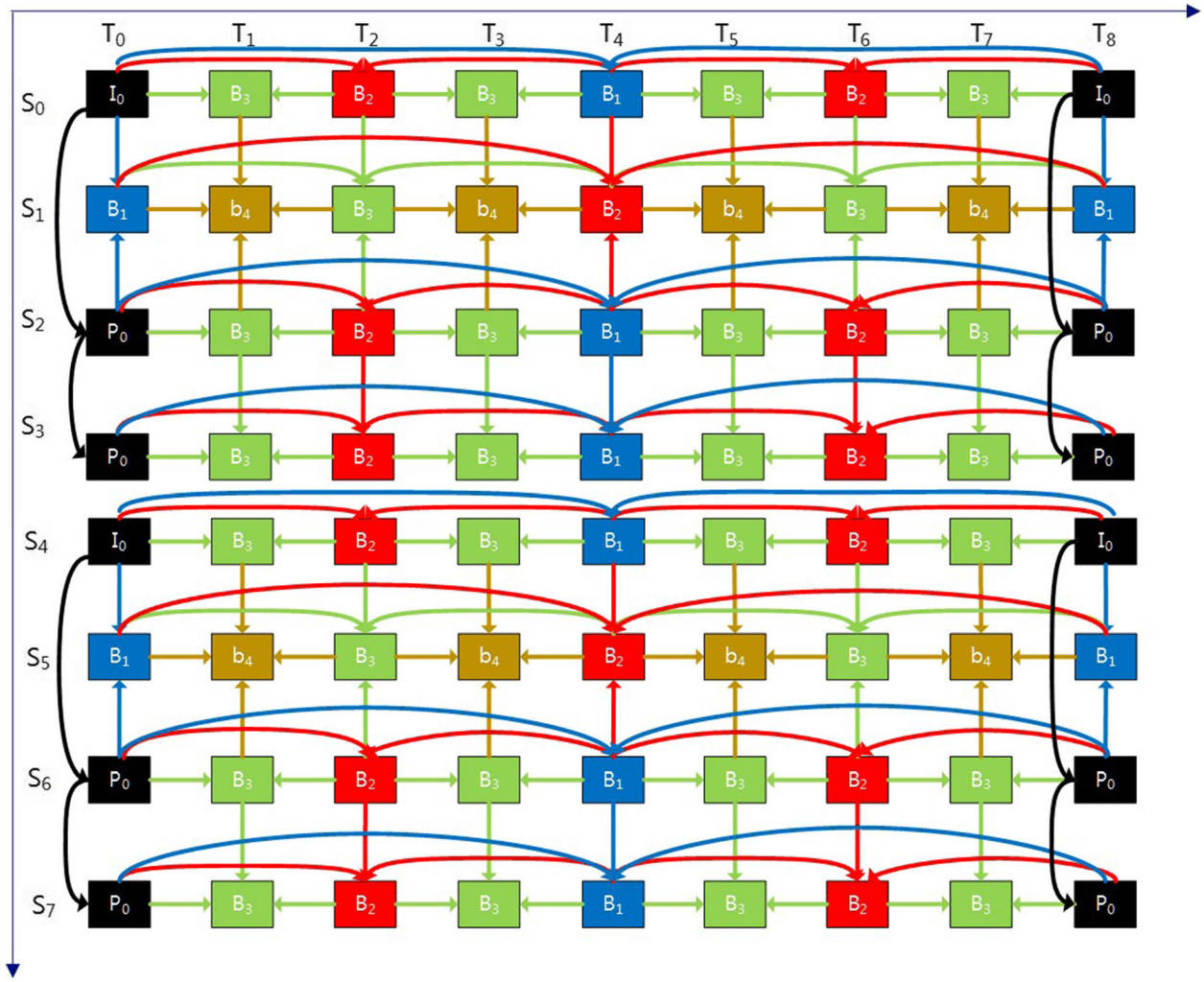

Fig. 8 Stereo stream ($N = 8$) in two independent sub-streams. In the Section 2

identification and use of this correlation will lead to a significant reduction of bit rate [4]. Similar to inter-view motion prediction, MV-HEVC uses advanced residual prediction (advanced residual Perdition, ARP), employing the residual code to predict the residual of the current block and transferring the difference between the two as a coding option. In multi-view video, there is a strong correlation between the different locations of the camera and the video of the same scene. Therefore, during the encoding process, the view images use the coding for the reference image motion estimation. With this method, only the high-level syntax elements need to be modified. The video images for the other views are added to the current reference list of the current coding image. This is the disparity compensation prediction (DCP) process. Also, in the multi-view video, the camera captures the same scene from different viewpoints. As the camera captures the same scene without calibration of the color transfer or lighting effect, the prediction may not be allowed due to lighting effects. Illumination compensation (IC) is used to solve this problem. IC is only used for the inter-view prediction, to compensate the brightness and chroma of the block to match the light of the current viewpoint and improve the prediction accuracy [2]. The other type of coding model is known as simulcast coding, where each frame is predicted only from frames of the same view. Simulcast has a simple prediction model in which there is no inter-prediction between each view. Therefore, with simulcast, all stream data can encode/decode independent of each other, and no additional processing is needed in a separate thread overhead. Consequently, each multi-view data can be encoded

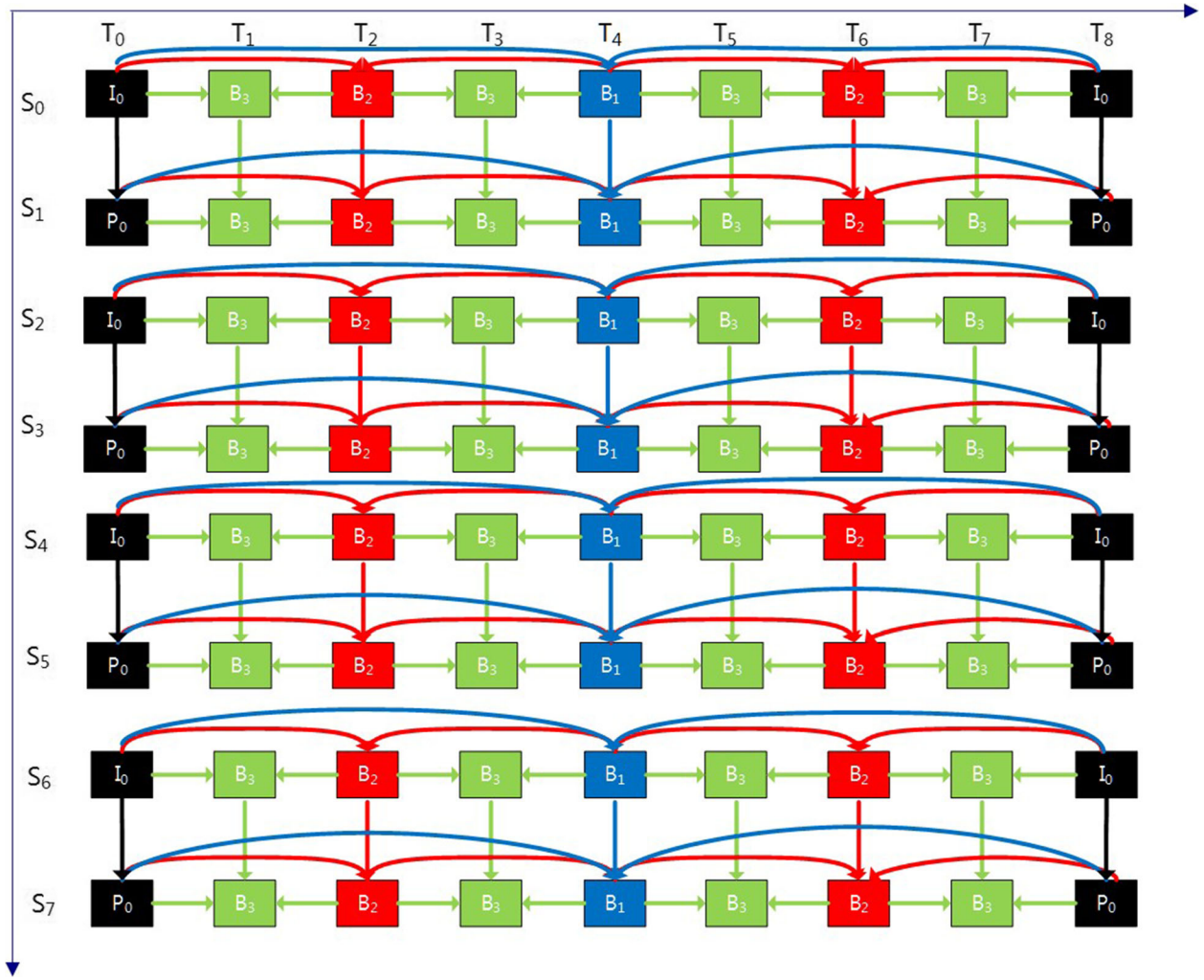

Fig. 9 Stereo stream ($N = 8$) in four independent sub-streams. In the Section 2

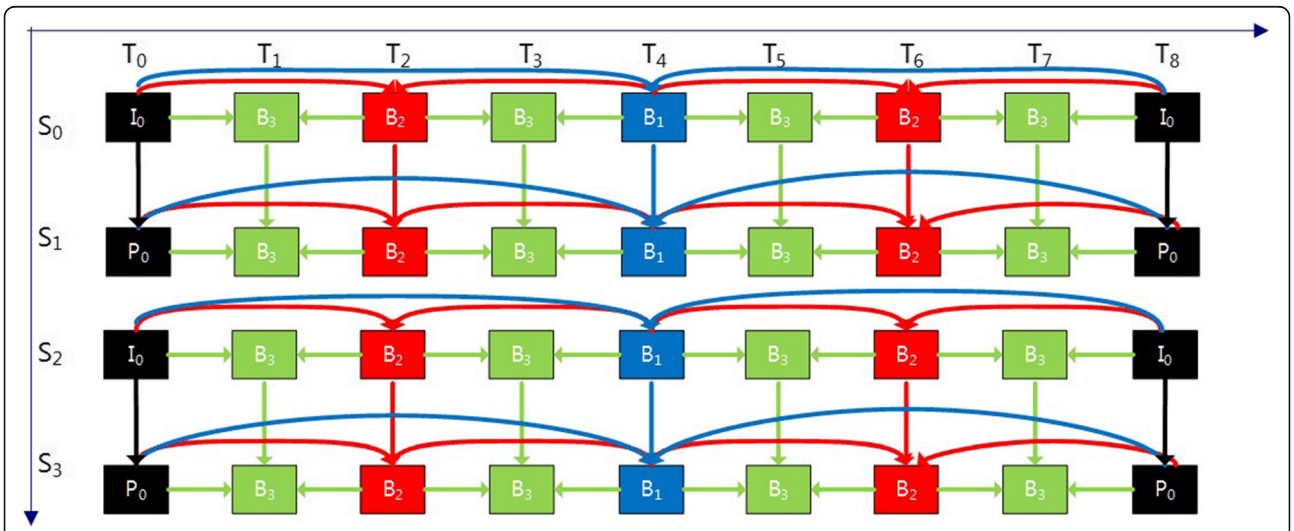

Fig. 10 Stereo stream ($N = 4$) in two independent sub-streams. In the Section 2

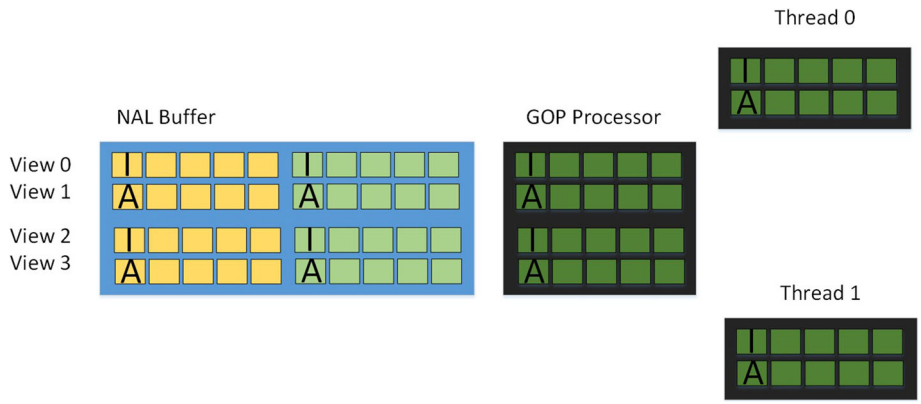

Fig. 11 Block diagram of the multi-threaded MV-HEVC. In the Section 2

and decoded separately by codecs with the HEVC standard. Although simulcast is simple, previous results show [18] that the method is inefficient compared to MV-HEVC. On the other hand, the full prediction scheme for the MV-HEVC structure is difficult to achieve with multi-threading. For the HEVC parallel processing architecture, researchers have put forward a lot of parallel recommendations and improvement methods, involving a higher degree of parallelism encoding and decoding algorithm to improve the coding speed. Examples include implementation of WPP parallel encoding and decoding based on slice-level parallel coding and cross-frame parallel coding based on WPP. These methods can improve the encoding and decoding speed, but the results are not great on the multi-thread utilization. We redesigned the encode/decode model, making it possible to use more efficient codecs in a multi-core platform.

2 Proposed method for MV-HEVC

In order to improve the decompression rate, we implement multi-core with multi-threaded processing. The proposed solution for multi-threaded processing is to decompose the input N view MV-HEVC stream to the M-independent sub-stream (one sub-stream implies one thread). Consequently, the new coding structure is a redesigned structure for encoding and decoding, in accordance with the MV-HEVC standard.

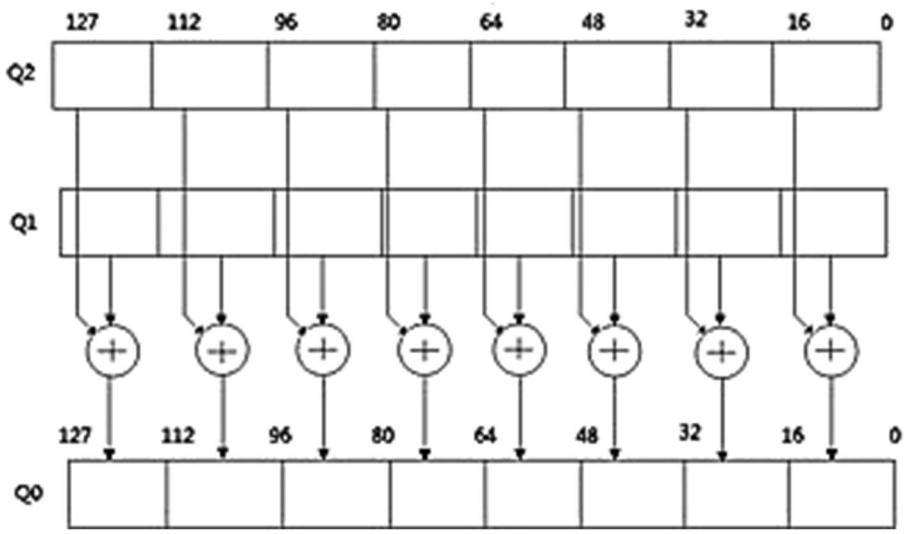

Fig. 12 ARM NEON architecture. In the Section 3

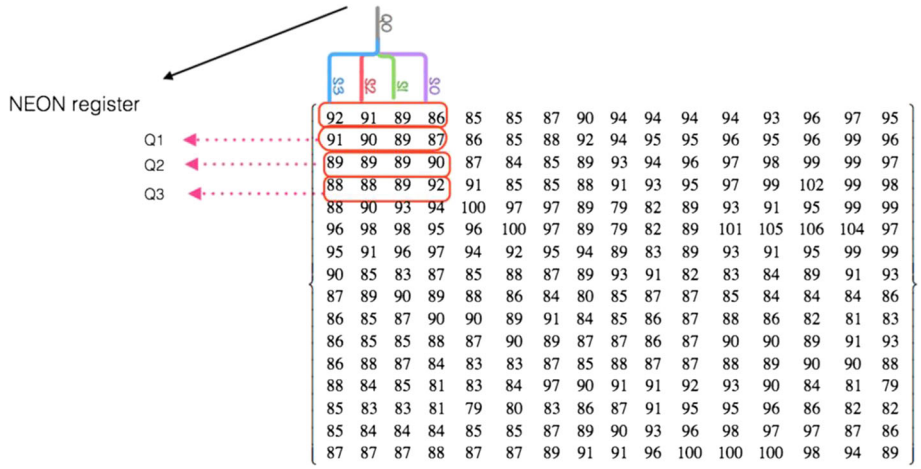

Fig. 13 Load TU block and IDCT coefficient into NEON register. In the Section 3

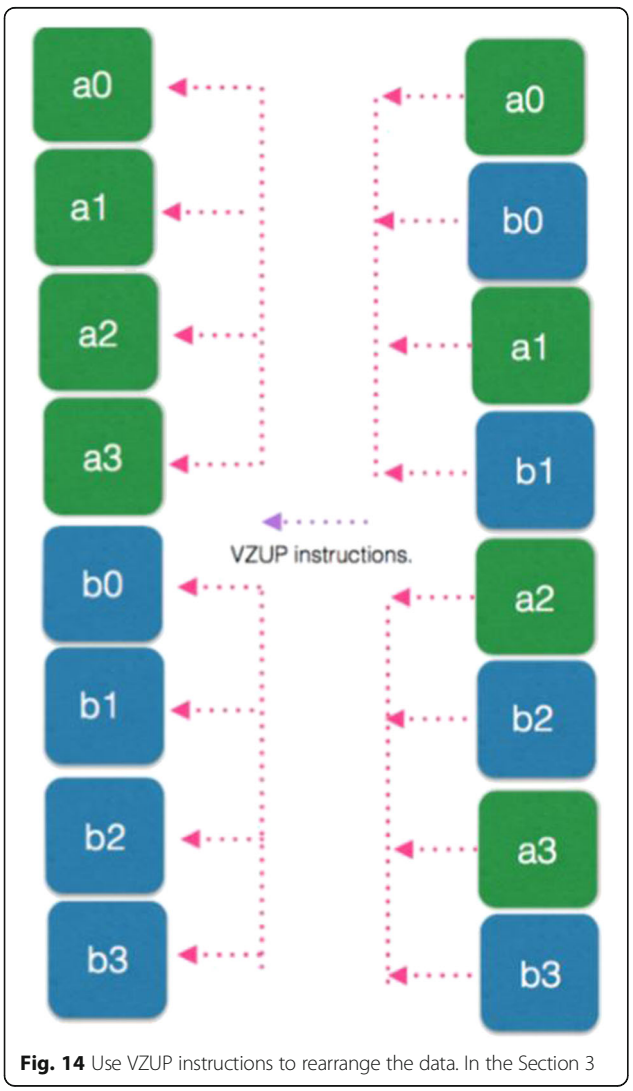

Fig. 14 Use VZUP instructions to rearrange the data. In the Section 3

2.1 Multi-core platform for propose method

We used the symmetric multiprocessing platform (SMP) shown in Fig. 5. This is the architecture used for ARM multi-core processing on the Linux system. The SMP architecture is a simple architecture for two or more identical processors that are connected via a shared memory. Each processor has equal access to the memory (the same access latency as the memory space). Typically, each processor has an associated private high-speed memory called a cache to accelerate the main memory data access and reduce the system bus traffic [19]. The mobile processor improves the performance by increasing the clock frequency. This method has a problem with increasing the power consumption and the subsequent increase of the clock frequency of the processor to build the multi-core architecture. Providing improved power efficiency while achieving higher performance, big.LITTLE architecture was introduced by ARM. In BIG/ITTLE architecture, the BIG and LITTLE cores have different characteristics. The BIG core provides higher performance, while the LITTLE core provides better power efficiency. The BIG core provides higher performance, but consumes more power, while the LITTLE core provides more power but has lower performance. The two cores have different modes of operation, including switching operation mode and HMP mode cluster. In cluster switching mode, the big cluster consists of identical big CPU cores such as Cortex-A15. The LITTLE cluster consists of identical LITTLE CPU cores such as Cortex-A7. All tasks are assigned to one cluster, and the other cluster is deactivated (Fig. 6) [6, 20] When the task load reaches a predefined workload threshold, all the tasks are switched to the core of the next cluster, while the previous cluster is disabled. HMP BIG/LITTLE mode is the most complex and flexible method for BIG/

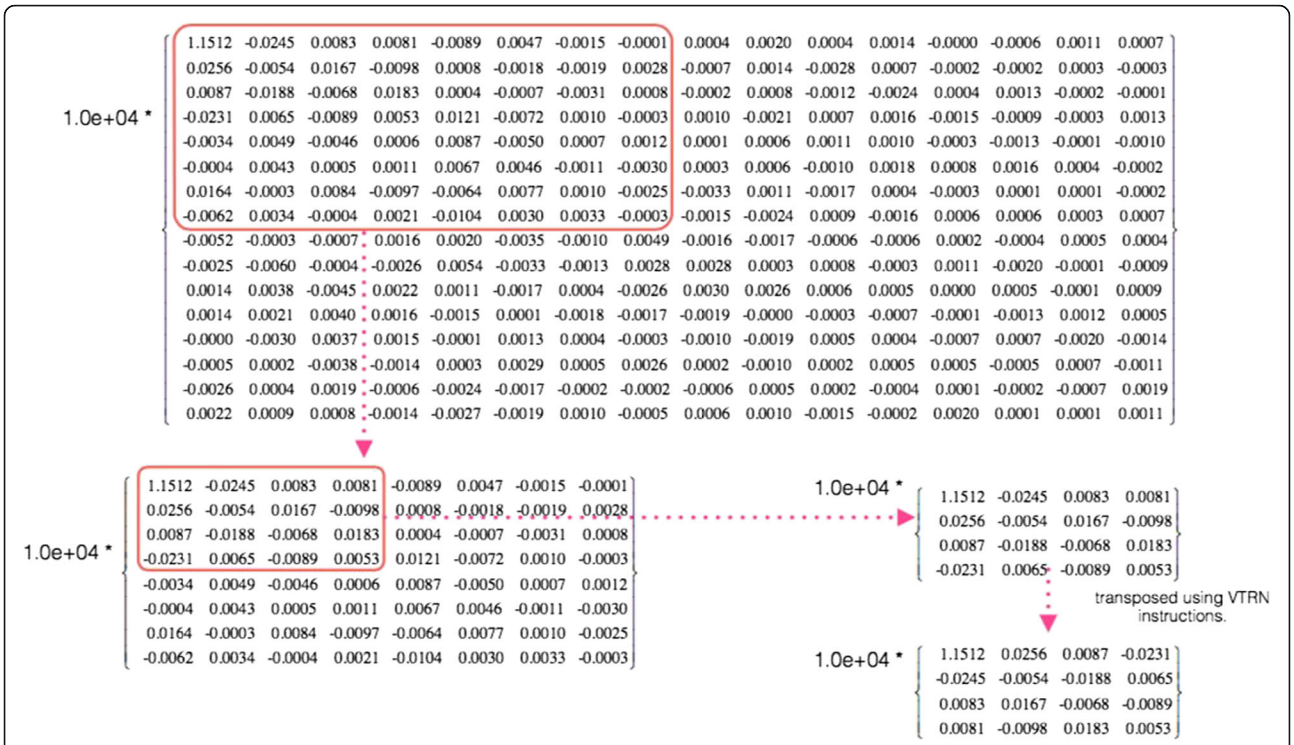

Fig. 15 The method of matrix partition. In the Section 3

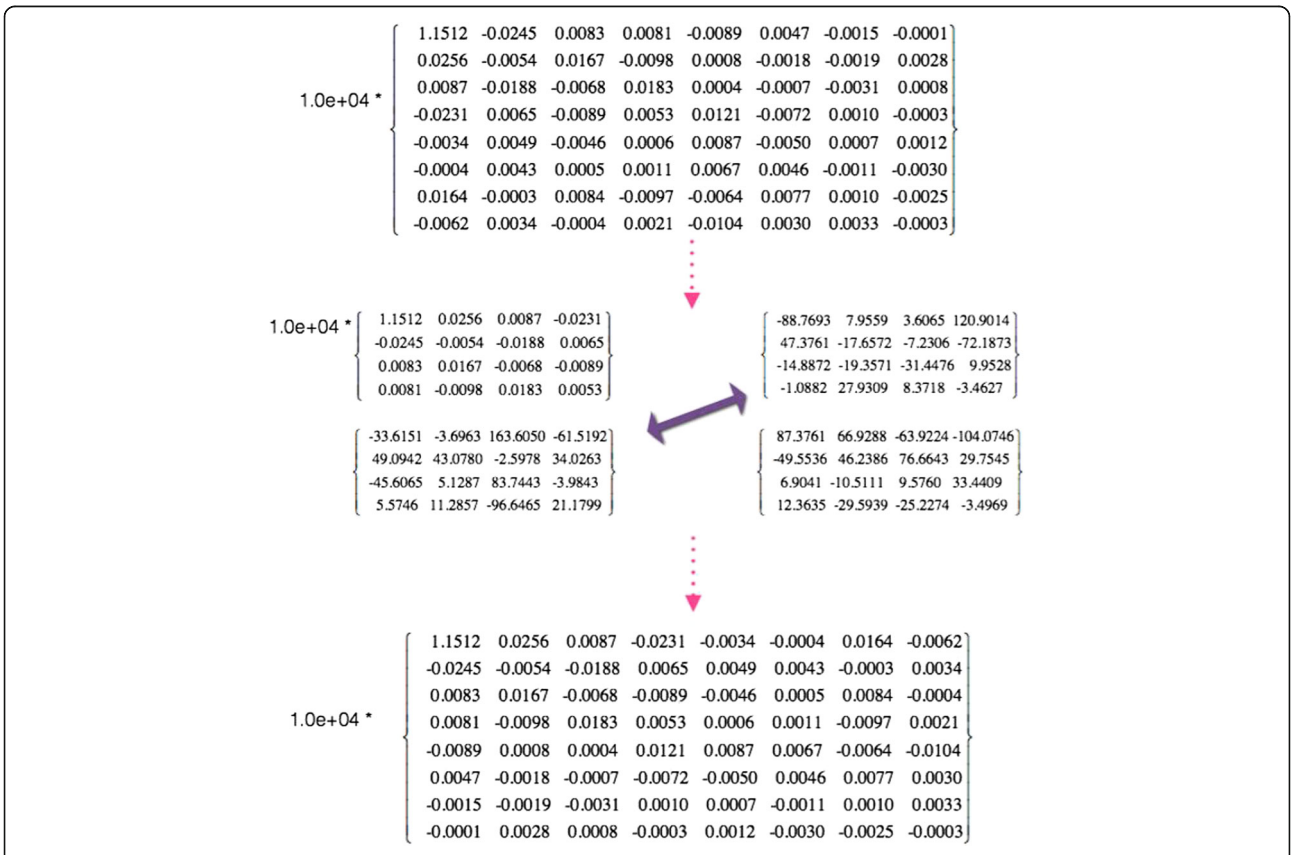

Fig. 16 Change the first 4 × 4 matrix and third 4 × 4 matrix. In the Section 3

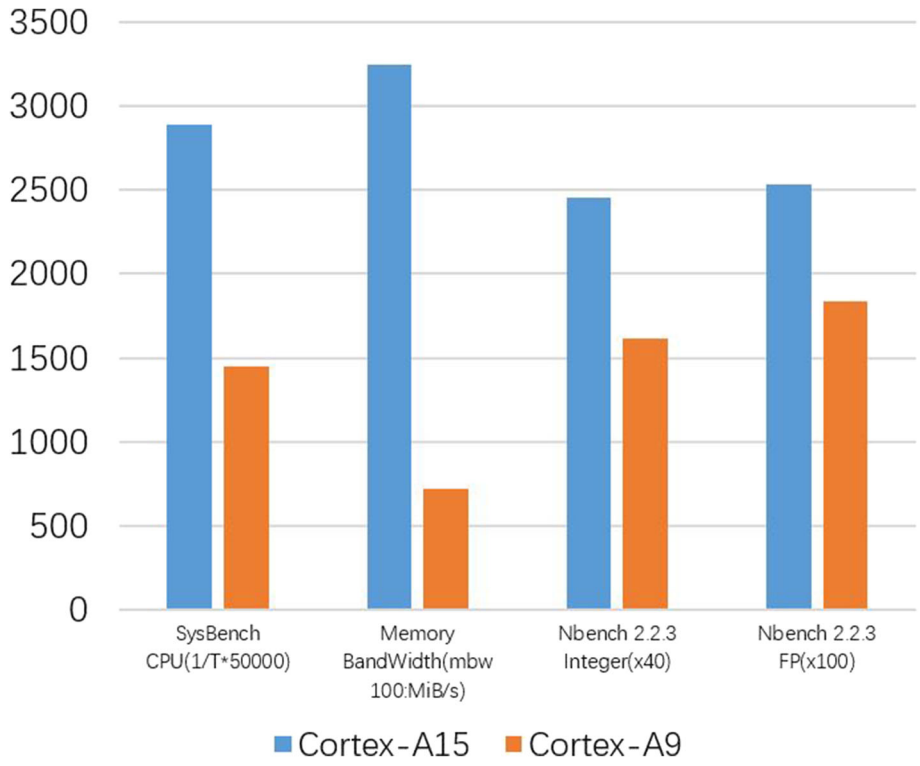

Fig. 17 Cortex-A15 Performance relative to Cortex-A9. In the Section 4

LITTLE architecture. In this mode, activation and deactivation are performed independently of each cluster, so that each CPU core can be independently activated or deactivated (Fig. 7) [20–22].

As described in Table 1, HMP solutions provide the highest flexibility and efficiency, performance, and power consumption. Compare Fig. 7 with Fig. 6 for example. Four big cores are activated in the cluster-switching mode in Fig. 6, while two big and two LITTLE cores are activated in the HMP mode with the same task characteristics as illustrated in Fig. 7. Therefore, the HMP model is the most energy-efficient and powerful solution for mobile CPUs [20–22]. We used the HMP mode SoC to implement our design in the big core (Cortex-A15) running a multi-thread MV-HEVC application and in the LITTLE core (Cortex-A7) running the operating system. This avoids the downside of the operating system taking up CPU resources. With the multi-threading API, we use the portable operating system interface thread, which can take full advantage of SMP [9], to optimize multi-threaded MV-HEVC. The POSIX thread provides the thread mechanism and shared memory. When a program is invoked that creates a number of threads, each thread provides its own stack (local variables and status). To support multi-threaded access to the shared memory, a coordination mechanism is needed. POSIX provides the mutex function to create a critical region by a thread with exclusive access to an object (a piece of memory) Fig. 5.

2.2 New multi-thread algorithm for MV-HEVC

We describe a new architecture for decoding MV-HEVC with N views on a target platform with M cores and

Table 2 Cortex-A9 and Cortex-A15 comparison

	Cortex-A9	Cortex-A15
Instruction set	ARMv7	ARMv7 (virtual 40b PA)
Core Config.	1, 2, 4	2, 4, 8 (4 × 2)
Speed per core (DMIPS/MHz)	2.5	3.5 to 4.01
L1 cache (KB)	32 + 32	32 + 32
L2 cache (MB)	1	Up to 4
Data Bus (bit)	32	32
SIMD Engine	ARM NEON (64 bit)	ARM NEON (128 bit)
Decoder width	2	3
Pipeline depth	8–11	15/17–25
FPU	VFPv3	VFPv4

Table 3 MV-HEVC decoding rates with ARM Cortex-A9 and Cortex-A15 at 1024 × 768 resolution

	One thread						Two threads						Four threads	
Processor	Cortex-A9			Cortex-A15			Cortex-A9			Cortex-A15			Cortex-A9	Cortex-A15
View num.	4	6	8	4	6	8	4	6	8	4	6	8	8	8
Balloon	4.32	3.33	–	11.41	8.28		8.58	6.61	–	22.29	17.16	–	–	–
Kendo	4.22	3.21	–	11.32	8.21		8.51	6.38	–	22.13	16.87	–	–	–
Newspaper	4.34	3.18	2.21	11.33	7.95	5.15	8.55	6.15	4.19	22.10	15.91	10.13	8.34	19.84

describe the implementation of a multi-threaded MV-HEVC client based on this architecture. The proposed solution for multi-threaded processing is to decompose the input N view MV-HEVC stream to the M-independent sub-stream. We proposed a solution with minimum loss of coding efficiency and minimum processing overhead, which refers to a combination of excess or indirect computation time, memory, or bandwidth. In a simulcast, all data streams can be coded independent of each other, with no processing overhead in separate threads. However, this leads to a loss of efficiency compared to the MV-HEVC. To address this problem, we improved the structure based on the MV-HEVC. For a multi-view video with 8 views, it is possible to generate two independently decodable sub-streams and zero to three videos can be viewed using the full prediction scheme of the MV-HEVC for coding insub-stream0. For four to seven views, the full prediction scheme of the MV-HEVC is used for coding in sub-stream1. However, between sub-stream0 and sub-stream1, simple prediction is used for the codec, so that each thread in the multi-view video is a full MV-HEVC prediction. As each thread is independent, there is no data sharing between threads as shown in Fig. 8. Therefore, there is no processing overhead. For the case of eight views run in four threads, four independently decodable sub-streams are required that can be generated by splitting each sub-stream of two threads. This is performed by defining the inter-view independencies between every thread, whereby views 0 and 1 use the full prediction scheme of the MV-HEVC as sub-stream0. The situation is the same for views 2–3, 4–5, and 6–7, which are run separately in sub-stream1, sub-stream2, and sub-stream3. Every sub-stream run in each thread is shown in Fig. 9. The prediction of the four views running in two threads is shown in Fig. 10. In order to use this proposed method described above, the coding method is conditional. In the case of eight views, four threads are needed, while the four-view multi-view video needs two threads. Multi-thread MV-HEVC generates M-independent decoded sub-streams that run in the MV-HEVC decoder in a separate thread from the received MV-HEVC stream. The block diagram of the multi-threaded MV-HEVC client is depicted in Fig. 11. The GOP processor receives NAL units for single GOP from the NAL buffer. It then generates the sub-streams and signals the threads for decoding. The results obtained by using this approach increased parallelization and reduced decoding time significantly. Thus, compared with single-thread decoding of eight views in MV-HEVC, our proposed two-thread method will reduce the decoding time consumed by about 50%. If four threads are used, the decoding time consumed will be reduced to almost 25%.

3 Proposed SIMD optimized MV-HEVC

After using multi-core processing to improve the decoding speed, however, for the 1024 × 768 MV-HEVC, real-time decoding is difficult. In this paper, we optimized the multi-view video coding decoder using ARM SIMD, commonly known as NEON [11, 12]. SIMD architecture supports easy data parallelization during computation. The design shown in Fig. 12 shows the architecture of an ARM A15 co-processor that supports an SIMD instruction set. Cortex-Ax processors support the NEON architecture. The NEON register

Table 4 MV-HEVC decoding rates with ARM Cortex-A9 and Cortex-A15 at 1920 × 1088 resolution

	One thread				Two threads				Four threads	
Processor	Cortex-A9		Cortex-A15		Cortex-A9		Cortex-A15		Cortex-A9	Cortex-A15
View num.	4	8	4	8	4	8	4	8	8	8
PoznanHall2	3.38	1.45	6.21	3.28	6.54	2.88	11.98	5.41	5.72	10.51
PonznanStreet	3.41	1.44	6.29	3.32	6.74	2.97	12.07	5.58	5.79	10.88
Undo_Dancer	3.39	1.43	6.22	3.29	6.71	2.80	12.01	5.46	5.52	10.84

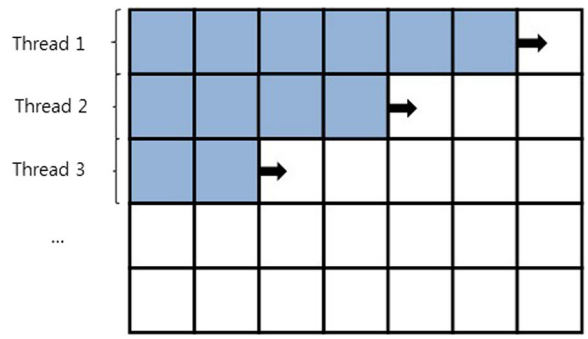

Fig. 18 Illustration of wavefront parallel processing. In the Section 4.2

file consists of 32 64-bits-wide registers, which can also be used as 16 128-bits registers. We apply the fast implementation of the HEVC decoder over ARM NEON processors.

The inverse transform method has always been a very suitable SIMD accelerated kernel. In HEVC, it is also true that the size of the HEVC block can be up to 32 × 32; hence, the transformation is more complex than with the previous standard. In the case of 4 × 4 IDCT, using the following step to accelerate. The 4 × 4 IDCT formula as follows

$$Y = C^T \cdot Z \cdot C$$

In the formula, Z is 4 × 4 TU block, C is the coefficient matrix of 4 × 4 IDCT, and Y is the final result of 4 × 4 IDCT. In the case of a 4 × 4 IDCT, we took the following steps: (1) We loaded the TU block and IDCT coefficient into the NEON register (Fig. 13). (2) We used a butterfly structure to compute the TU data and IDCT coefficients. (3) Because the data obtained was not in the same register, we used the Vector Unzip (VZUP) instructions to rearrange the data, as shown in Fig. 14. (4) If the block size of the TU was larger than 4 × 4, the matrix was divided, as shown in Fig. 15.

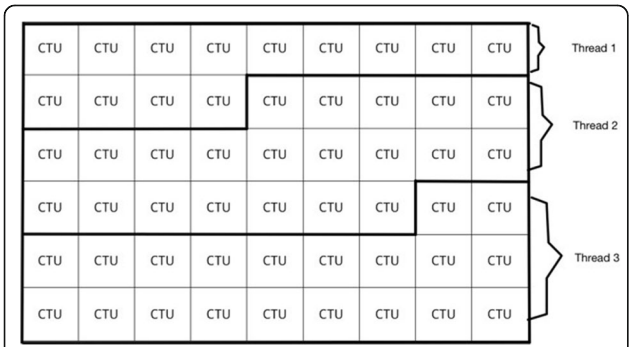

Fig. 19 Illustration of slice. In the Section 4.2

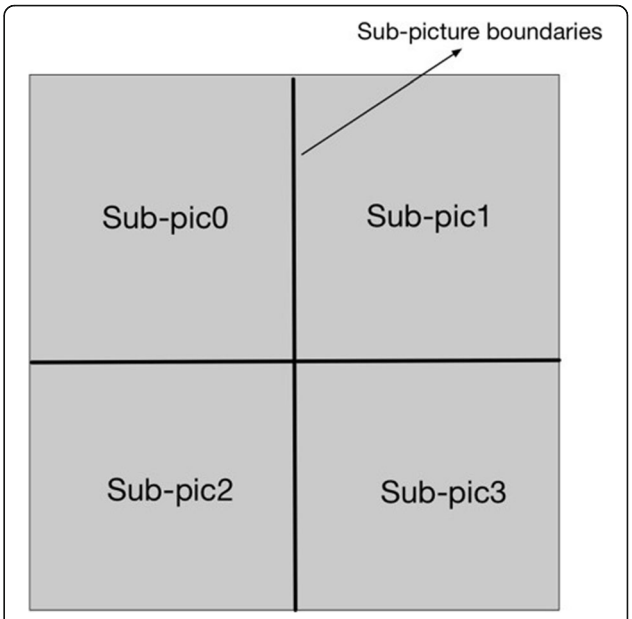

Fig. 20 Illustration of Tile. In the Section 4.2

(5) After repeating the first three steps, a number of 4 × 4 matrices were obtained. If the transformation of the TU block resulted in the 8 × 8 matrix as shown in Fig. 16, we had to change the first and third 4 × 4 matrices.

4 Experimental result and discussion

In order to ensure that we have an optimized MV-HEVC decoder, we modified HTM-15.0, which is MV-

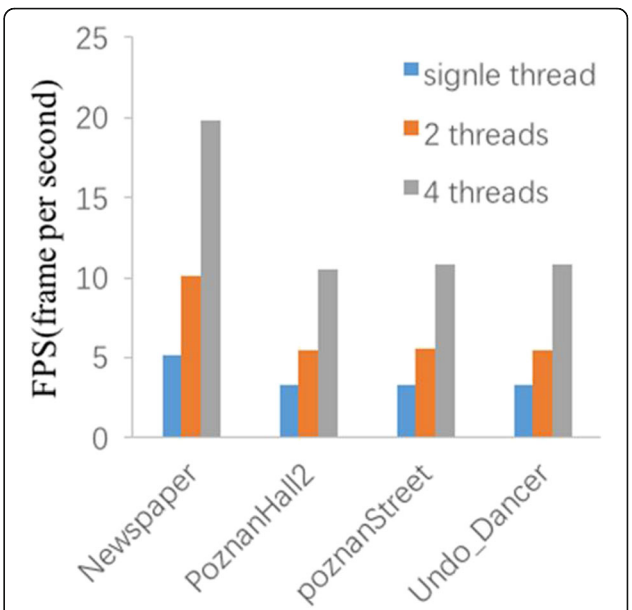

Fig. 21 Comparison of each thread decoding rate. In the Section 4.2

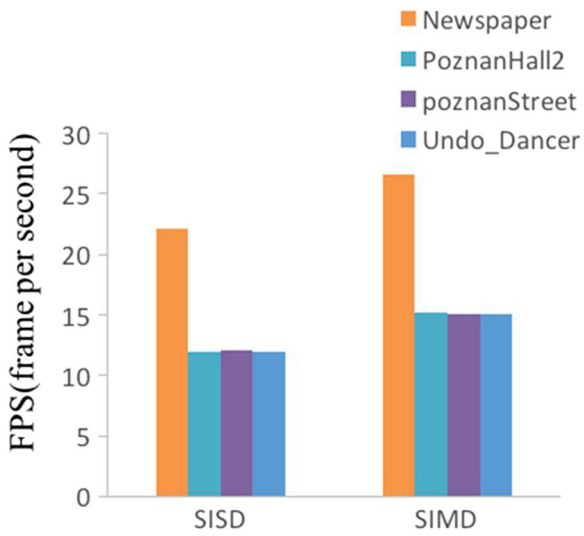

Fig. 22 Comparison of SIMD with SISD decoding rate. In the Section 4.2

HEVC reference software. For both platforms, the GCC 4.8.1 compiler and the −O3 optimization level were used. In SIMD optimization, we added the -mfpu = neon compiler option. The execution time is measured outside of the program, using the time command. We test the ARM Cortex-A9 and Cortext-A15 single, and the dual and quad-core core. The frequencies of both processors are 1.4 GHz for the Cortex-A9 and 2.1 GHz for Cortex-A15. ARM architecture can achieve an effective balance of high performance, low power, and small size [23]. The instruction set for ARM Cortex-A9 and Cortex-A15 architectures is ARMv7. As indicated in Fig. 17, the performance of the Cortex-A9 and Cortex-A15 cores differ by a factor of 1.5 ~ 2 times based on the core integer [24, 25]. The differences between the Cortex-A9 and Cortex-A15 can be seen in Table 2. We tested four different multi-view views, including two, four, six, and eight views. We also tested two resolutions: 1024 × 768 and 1920 × 1080 [19].

4.1 Performance optimized with multi-threaded

For the multi-threaded processing strategy introduced above, threads decode the deepened thread number and multi-view video views the number. Tables 3 and 4 present the performance of the decoding frame rate in the ARM Cortex-A9 and ARM Cortex-A15 processors. One thread means that it is a default module while two threads and four threads imply optimization with two threads and four threads. We tested the 1024 × 768 and 1920 × 1088 multi-videos. The balloon, Kendo, and Newspaper videos all have a resolution of 1024 × 768 whereas the PoznanHall2, Poznan Street, and Undo_Dancer videos have a resolution of 1920 × 1088. The balloon and Kendo test video sequences are recorded on seven cameras so they have a maximum of seven views, but the redesigned multi-threaded MV-HEVC using four threads for decoding needs at least eight views for the codec. Significant performance gains from the proposed architectures are used in ARM multi-core platforms. The decoding rate for all cases is improved almost linearly with the number of threads. The results are shown in Chart 2, which indicates that the decoding performance is affected by the number of views and the test sequence. Overall, the decoder scales well with multi-threaded processes for both 1980 × 1088 and 1024 × 768. With two threads, the speed-up is around 1.85 times, while with four threads, the speed-up is around 3.85 times. We use Amdahl's law found that the theoretical speed-up is around four times; Gene Amdahl, a computer architect and IBM fellow, developed computer architectures [26]. Gene Amdahl is best known for his method of predicting the maximum improvement of a system when the system is partially improved [26] using multiple processors. Amdahl's law shows the most commonly used calculation for the maximum theoretical performance improvement. Where N is the number of processors and F represents the portion of the system that cannot be parallelized (the portion of the system that is sequential in nature) [26, 27]. Using the equation of Amdahl's law, the maximum performance of a parallel processing system can be calculated because there are no serial parts in our proposed method. Therefore, the F is equal to 0, N is 4, and Speedup is 4, but as expected, the experiment rate is slower than the theoretical rate. We can identify this as the reason why the throughput of the multi-core processors is not exactly linear with the number of cores and this creates an upper bound. Another reason is that the amount of MV-HEVC data is too large for the memory requirements, although the ARM processors

Table 5 MV-HEVC decoding rates with SIMD optimized (ARM Cortex-A9and ARM Cortex-A15 at 1024 × 768 resolution)

	One thread						Two threads						Four threads	
Processor	Cortex-A9			Cortex-A15			Cortex-A9			Cortex-A15			Cortex-A9	Cortex-A15
View num.	4	6	8	4	6	8	4	6	8	4	6	8	8	8
Balloon	5.51	4.11	–	13.71	10.63	–	10.67	7.89	–	26.82	20.63	–	–	–
Kendo	5.42	3.95	–	13.61	10.51	–	10.35	7.68	–	26.63	20.34	–	–	–
Newspaper	5.39	3.85	2.69	13.60	10.11	6.42	10.38	7.41	5.22	26.58	19.24	12.34	10.20	23.85

Table 6 MV-HEVC decoding rates with SIMD optimized (ARM Cortex-A9 and ARM Cortex-A15 at 1920 × 1088 resolution)

	One thread				Two threads				Four threads	
Processor	Cortex-A9		Cortex-A15		Cortex-A9		Cortex-A15		Cortex-A9	Cortex-A15
View Num.	4	8	4	8	4	8	4	8	8	8
PoznanHall2	4.17	1.86	7.85	4.01	8.16	3.63	15.23	6.74	7.09	13.11
PonznanStreet	4.36	1.85	7.81	4.08	8.43	3.64	15.06	6.91	7.15	13.41
Undo_Dancer	4.29	1.81	7.72	4.03	8.40	3.51	15.01–	6.76	6.89	13.37

are known to be more memory limited. Although the thread mutex lock is not used, synchronization and thread creation have an overhead.

$$\text{Speed up} = \frac{1}{F + (1-F)/N}$$

Amdahl's law for processor parallelization

4.2 Discuss other multi-threaded

The authors [28] propose a method called WPP (wavefront parallel processing), which uses parallel operations to improve the coding performance. The WPP solution is that splits the frame into CTU rows. As illustrated in Fig. 18, the first row is processed in an ordinary manner, and the second row can only be processed after two CTUs have been processed in the first row. The third row can only be processed after two CTUs have been processed in the second row and so on. The CABAC context at each CTU row is initialized by the CABAC context state at the second CTU of the previous CTU row [2]. The operation problem is that multi-core utilization efficiency is low. When using WPP, every row should wait for two CTUs encoding/decoding of the previous row for CABAC context initialization; thereby, this delay propagates as number of CTU rows executing simultaneously on cores grows. In contrast, when increasing the number of cores, we propose a method that has no such problem.

In addition, the authors [29] propose slice-level parallel scheme. Slices are a sequence of CTUs that are processed in the order of a raster scan. A picture may be split into one or several slices as shown in Fig. 19, so that a picture is a collection of one or more slices [30]. The problem with slice in parallel computing is that the number of CTU on the thread is different. Thus causes a thread which has most of the CTU to code a long time.

The authors [31] propose a method called Tile, which is advantageous for parallel decoding purpose when compared to slices or WPPs. In the example shown in Fig. 20, a picture is divided into four sub-pictures of equal size. With tiles, the picture can be divided into 2 × 2 sub-pictures. All Tiles are independent of each other and the threads are uniformly distributed but the de-blocking filter, sample adaptive offset (SAO), adaptive loop filter (ALF) can cross tile boundaries. Relatively speaking, our proposed method in the parallel processing includes the ALF and SAO.

Motion estimation is the most time-consuming step in HEVC interframe coding. The authors [32] used multi-core CPU and GPU platforms to achieve the coding of parallel processing. The encoder is divided into six modules, and one of the most time-consuming motion estimation module codec is processed by the GPU. Also, the author [33] proposed motion estimation in parallel processing with GPU implementation. All PU of the CTU occurs first through GPU pre-processing. The MV

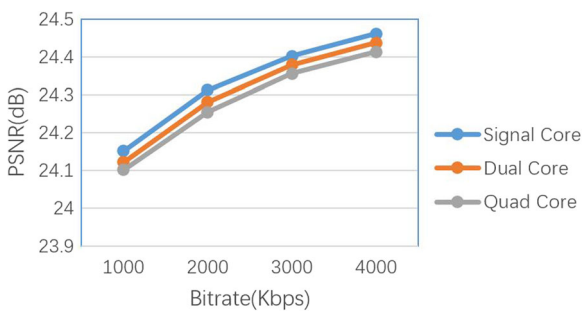

Fig. 23 Rate-distortion performance of MV-HEVC standard for balloon. In the Section 4.3

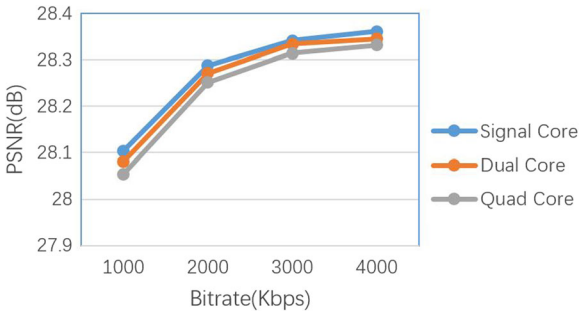

Fig. 24 Rate-distortion performance of MV-HEVC standard for Kendo. In the Section 4.3

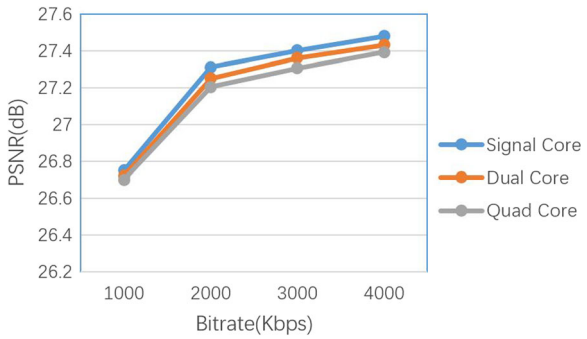

Fig. 25 Rate-distortion performance of MV-HEVC standard for Newspaper. In the Section 4.3

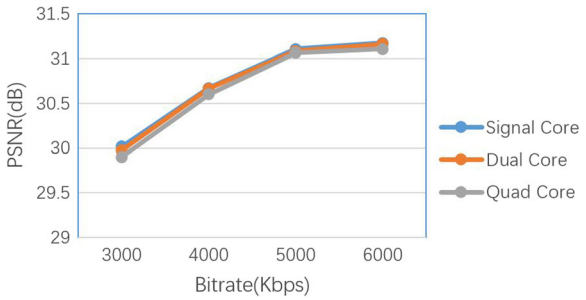

Fig. 27 Rate-distortion performance of MV-HEVC standard for poznanHall. In the Section 4.3

and the corresponding value are obtained, then the optimal MV is determined. This algorithm can not only perform parallel motion estimation for all PU in the same CTU but also be processed simultaneously by different CTU. Therefore, the speed increase is very significant as shown in Figs. 21 and 22. The parallel scheme proposed by the authors is based on the GPU co-design. Our proposed method can also be carried out with acceleration motion compensation and motion estimation with GPU, because in our proposed method, each thread codec is an independent MV-HEVC

4.3 Performance optimized with ARM NEON

With the SIMD acceleration, Tables 5 and 6 present the result of decoding rates in ARM Cortex-A9 and Cortex-A15 processors. Clearly, the performance gains from the SIMD is 15fps in four views with two threads at 1920 × 1088 resolution and around 26.5fps in four views with two threads at 1024 × 768. Chart 3 shows that the optimized SIMD can increase the decoding frame rate by around 1.25 times that of SISD. Gene Amdahl, a computer architect and IBM fellow, developed computer architectures [26]. Gene Amdahl is best known for his method of predicting the maximum improvement of a system when the system is partially improved [26] using multiple processors. Amdahl's law shows the most commonly used calculation for the maximum theoretical performance improvement.

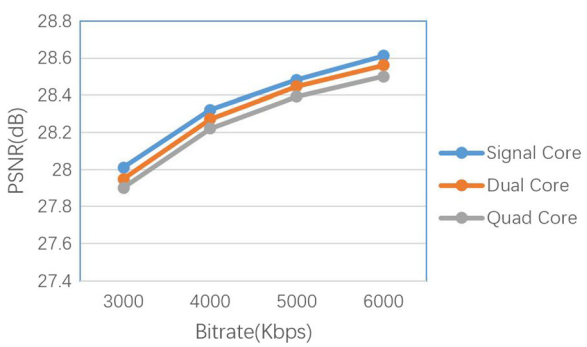

Fig. 26 Rate-distortion performance of MV-HEVC standard for Undo_dancer. In the Section 4.3

Figures 23, 24, 25, 26, and 27 display the resulting weighted PSNR (0.75 × Y-PSNR + 0.125 × U-PSNR + 0.125 × V-PSNR) for all the decoded videos with multi-threaded and SIMD optimization in quantization parameter (QP) 45. The results indicate that at a similar bitrate, with an increase of the number of threads, the peak signal-to-noise ratio (PSNR) reduced slightly. However, the human eye does not detect such a subtle change; hence, the multi-threading and SIMD optimization has minimal effect on image quality.

4.4 Discussion

Table 7 gives the individual evaluation results of the proposed algorithm compared with the original MV-HTM slice multi-threading. The Bjontegaard delta PSNR (BDPSNR) [34] represents the average PSNR gain; bitrate (BDBR) represents the improvement of total bitrates for multi-view video coding. The results show the proposed method relative to the MV-HTM 15.0 with slice multi-threading, the average BDPSNR decreased by 0.063 BDBR, and the average decibel (dB) only increased by 1.4%. The average decoding time decreased by 41.55%. In HEVC multi-threading methods, Tile is the fastest, but we found that, although Tile could improve the video decoding rate, it could not meet the real-time decoding requirements. Table 8 shows a comparison

Table 7 Comparison results of HTM

	BDBR(%)	BDPSNR(%)	ΔT(%)
Blloon	2.19	−0.056	−43.08
Kendo	0.68	−0.033	−41.31
Newspaper	1.36	−0.024	−40.32
Undo_dancer	1.7	−0.025	−41.85
poznanHall	1.07	−0.077	−45.21
Average	1.4	−0.063	−41.96

Table 8 Comparison results of the proposed method with Tiles (ARM Cortex-A15)

	Two threads						Four threads	
Processor	Tiles			Propose method			Tiles	Propose method
View num.	4	6	8	4	6	8	8	8
Balloon	24.43	18.76	–	26.82	20.63	–	–	–
Kendo	24.13	18.56	–	26.63	20.34	–	–	–
Newspaper	24.08	18.52	12.01	26.58	19.24	12.34	22.17	23.85

throughput of our proposed method with Tile in ARM Cortex-A15 with two and four threads. Result shows that the proposed method can increase the decoding frame rate by around 1.09 and 1.07 times that of Tiles with two and four threads. The reason for the implementation in the ARM processor is compared to other processors, ARM supports better scalability and portability at the application level. In the scalability, the proposed method can add GPU or FPGA as co-processor to further accelerate, because of video coding have large number of matrix operations; therefore, many optimizations can be implemented in GPU. However, our proposed multi-threading method only applies to mv-hevc or 3d-hevc not include HEVC or other extension standards of HEVC.

5 Conclusions

In this paper, we proposed an optimized method for MV-HEVC using multi-threading and SIMD instructions implemented on ARM processors. Based on the results, the proposed implementation of four threads and SIMD instructions was found to be around 4.8 times faster than that of the HEVC reference software, HTM 15.0. Although tile and slice presents a parallel optimization approach for MV-HEVC and actually result in good processing speeds on X86/X64, it was found to be slower on ARM processors. On the other hand, the proposed optimization method of MV-HEVC showed significant improvement in terms of processing speed on ARM processor mobile platforms. With the promotion of the next generation video coding standard, MV-HEVC, and the increasing number of mobile terminals, multi-view video can be watched in real time at a resolution of 1024×768.

Acknowledgements
This work was supported by the Konkuk University in 2015.

Authors' contributions
LW proposed the framework of this work, carried out the whole experiments, and drafted the manuscript. YB C offered useful suggestions and helped to modify the manuscript. LJ participated in its design and coordination and helped to draft the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Authors' information
LW received M.S. degree from Konkuk University (2016) and a Bachelor's in Technology degree in Automation from Shenyang Ligong University, Shenyang, China, in 2014. He is currently working toward a Ph.D. degree in Electronics Engineering at Konkuk University in Seoul, South Korea. His research interests include system-on-chip (SoC) design, image processing, multi-core processor systems.
LJ received a Bachelor's in Technology degree in Electronics Engineering from Konkuk University, Seoul, South Korea, in 2015. She is currently working toward a Master's degree in Electronics Engineering at Konkuk University in Seoul, South Korea. Her research interests include hardware-software co-design of video processing, embedded software, System-on-Chip (SoC) design, and design of multi-core processor systems.
YBC (M'86) received a B.Sc. degree from Kyongbuk University (1981), a M.Sc. degree from the Univ. of S. Carolina (1988) and a Ph.D. degree from Case Western Reserve University, OH, USA(1992). He is currently a professor in the Department of Electronics Engineering at Konkuk University, Seoul, Korea. His research interests include embedded system design, SoC design, networking systems, application of image processing to mobile environments, and digital communication system design for mobile and ad-hoc networks.

Received: 22 July 2016 Accepted: 28 February 2017
Published online: 20 March 2017

References

1. ITU-T rec, *High Efficiency Video Coding, document Rec. ITU-T H.265 and ISO/IEC 23008-2*, 2013
2. GJ Sullivan, JR Ohm, H Woo-Jin, T Wiegand, Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology **22**(12), 1649–1668 (2012)
3. ITU, *H.265 : High efficiency video coding*, 2015
4. GJ Sullivan, JM Boyce, C Ying, JR Ohm, CA Segall, A Vetro, Standarized extensions of High Efficiency Video Coding (HEVC). IEEE Journal of Selected Topics in Signal Processing. **7**(6), 1001–1016 (2013)
5. Sze V, Budagavi M, High throughput CABAC entropy coding in HEVC, IEEE Trans. Circuits Syst. Video Technol. **22**(12). 1778–1791 (2012)
6. A Norkin, G Bjontegaard, A Fuldseth et al., HEVC deblocking filter. IEEE Trans. Circuits and Syst. Video Technol. **22**(12), 1746–1754 (2012)
7. K Tech, Y Wegner, M Chen, J Hannuksela, *Boyce, MV-HEVC Draft Text 9, in document JCT3V-I1002, Sapporo, JP*, 2014
8. J Boyce, Y Chen, D Chen, MM Flynn, M Hannuksela, C Naccari, K Rosewarne, J Sharman, GJ Sole, T Sullivan, G Suzuki, YK Tech, K Wang, Y Wegner, *Ye Edition Text of High Efficiency Video Coding (HEVC), Including Format Range (RExt), Scalability (SHVC), and Multi-View (MV-HEVC) Extensions 2 Draft, in document JCTVC-R1013, Sapporo. JP*, 2014
9. Pthreads Tutorial. https://computing.llnl.gov/tutorials/pthreads/. Accessed 15 Dec 2014.
10. M Domański, T Grajek, D Karwowski, J Konieczny, M Kurc, A Łuczak, R Ratajczak, J Siast, O Stankiewicz, J Stankowski, K Wegner, *Coding of multiple video+depth using HEVC technology and reduced representations of side views and depth maps* (Picture Coding Symposium (PCS), Krakow, 2012), pp. 5–8
11. ARM NEON. http://www.arm.com/products/processors/technologies/neon.php. Accessed 19 Feb 2015.
12. Programmers model for NEON and VFP unit. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0438i/CDECGBDJ.html. Accessed 25 Oct 2015.

13. M Domański, T Grajek, D Karwowski, K Klimaszewski, J Konieczny, M Kurc, A uczak, R Ratajczak, J Siast, O Stankiewicz, J Stankowski, K Wegner, *Multiview HEVC – experimental results JCT-VC (MPEG/VCEG) Doc. JCTVC- G582, Geneva*, 2011
14. Y Chen, Y-K Wang, K Ugur, M Hannuksela, J Lainema, M Gab-bouj, The emerging MVC standard for 3D video services. EURASIP J. Adv. Signal Process **2009**, 1 (2009)
15. A Vetro, T Wiegand, GJ Sullivan, Overview of the Stereo and Multiview Video Coding Extensions of the H.264/MPEG-4 AVC Standard. Proceedings of the IEEE. **99**(4), 626–642 (2011)
16. J Stankowski, M Domanski, O Stankiewicz, J Konieczny, J Siast, K Wegner, Extensions of the HEVC technology for efficient multiview video coding, image processing (ICIP). 19th IEEE International Conference on, Orlando F. **2012**, 225–228 (2012)
17. K Ugur, H Liu, J Lainema, M Gabbouj, H Li, *Parallel Encoding - Decoding Operation for Multiview Video Coding with High Coding Efficiency, 3DTV Conference, 2007, Kos Island*, 2007, pp. 1–4
18. CG Gurler, A Aksay, GB Akar, AM Tekalp, *Multi-threaded architectures and benchmark tests for real-time multi-view video decoding, Multimedia and Expo, 2009. ICME 2009* (IEEE International Conference on, New York, 2009), pp. 237–240
19. LJ Karam, I AlKamal, A Gatherer, GA Frantz, DV Anderson, BL Evans, Trends in multicore DSP platforms. Signal Processing Magazine, IEEE. **26**(6), 38–49 (2009)
20. big.LITTLE Technology Moves Towards Fully Heterogeneous Global Task Scheduling, ARM. https://www.arm.com/files/pdf/big_LITTLE_technology_moves_towards_fully_heterogeneous_Global_Task_Scheduling.pdf.
21. big.LITTLE Technology: The Future of Mobile, ARM. https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf. Accessed 20 July 2015.
22. Hongsuk Chung, Munsik Kang, Hyun-Duk Cho, Heterogeneous Multi-Processing Solution of Exynos 5 Octa with ARM® big.LITTLETM Technology, SAMSUNG. https://www.arm.com/files/pdf/Heterogeneous_Multi_Processing_Solution_of_Exynos_5_Octa_with_ARM_bigLITTLE_Technology.pdf. Accessed 8 Nov 2015.
23. ARM Architecture Reference Manual, *ARMv7-A and ARMv7-R ed* (ARM Holdings, Cambridge, 2014), p. A1-28 - 1-35
24. Cortex-A15 Processor. http://www.arm.com/products/processors/cortex-a/cortex-a15.php. Accessed 6 July 2015.
25. ARM Technical Reference Manual, *The ARM Cortex-A9 Processors* (ARM Holdings, Cambridge, 2014)
26. GM Amdahl, *Validity of the single-processor approach to achieving, large scale computing capabilities, AFIPS Conference Proceedings, vol. 30 (Atlantic City, N.J. Apr. 18-20)* (AFIPS, Reston, 1967), pp. 483–485
27. M Tim Jones, *Linux and symmetric multiprocessing* (IBM, 2007), p. 14
28. C Gordon, F Henry, S Pateux, *JCTVC-F274: Wavefront Parallel Processing for HEVC Encoding and Decoding, Joint Collaborative Team on Video Coding (JCTVC)*, 2011
29. K Misra, J Zhao, A Segall, *JCTVC-C256: New results for entropy slices for highly parallel coding, Joint Collaborative Team on Video Coding (JCTVC)*, 2010
30. W Hamidouche, M Raulet, O Deforges, 4K real-time and parallel software video decoder for multilayer HEVC extensions. IEEE Transactions on Circuits and Systems for Video Technology. **26**(1), 169–180 (2016)
31. K Misra, A Segall, M Horowitz, S Xu, A Fuldseth, M Zhou, An overview of tiles in HEVC. IEEE Journal of Selected Topics in Signal Processing **7**(6), 969–977 (2013)
32. X Wang, L Song, M Chen et al., *Paralleling variable block size motion estimation of HEVC on CPU plus GPU platform, International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013
33. S Radicke, J Hahn, C Grecos et al., *A highly-parallel approach on motion estimation for high efficiency video coding (HEVC), International Conference on Consumer Electronics (ICCE)*, 2014
34. G Bjøntegaard, *Calculation of average PSNR differences between RD-curves, ITU-T Q.6/SG16 VCEG 13th Meeting, Document VCEG-M33, Austin, USA*, 2001