High Efficiency Video Decoding on Multicore Processor

Hyeonggeon Lee\textsuperscript{1}, Jong Kang Park\textsuperscript{2}, and Jong Tae Kim\textsuperscript{1,2}
Department of IT Convergence\textsuperscript{1}
Sungkyunkwan University
Suwon, Korea
Department of Electrical and Computer Engineering\textsuperscript{2}
Sungkyunkwan University
Suwon, Korea

Abstract—In this paper we present a High Efficiency Video Coding (HEVC) decoder implemented using multicore processor. HEVC can support Ultra High Definition (UHD) digital TV and resolution up to 8192x4320. It aims to achieve compression rate in the range of 50% bit-rate relative to existing standards. And decoding speed should be over the 30 fps (frames per second). Although multicore processors have sufficient performance and enough memory, HEVC software cannot make use it very well. So not only the proper architecture for HEVC is needed but also modified HEVC software should be considered. Gem5 simulator was used to simulate the performance of our decoder. Based on the simulation results we suggest appropriate multicore architecture platform which can satisfy the goal of HEVC decoding speed.

Keywords: HEVC, Multicore, Parallelization

1 Introduction

In recent days, as digital display technology has remarkably developed and high-definition of digital TV is needed for various media resources, Joint Collaborative Team on Video Coding (JCT-VC) which ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) have established develops the High Efficiency Video Coding (HEVC) standard. HEVC can handle the 8 K Ultra High Definition (UHD) and resolution up to 8192x4320. And it improves the data compression ratio compared to H.265/MPEG-4 AVC which is pre-version of HEVC. H.264/MPEG-4-AVC can be performed in general computer enough. But HEVC controls more fragmented unit for high definition video resources and adopts technologies to accomplish the parallelization tasks. So, new platform architecture which can realize the encoding and decoding these feature of HEVC tasks is needed. But there are so many possible architecture components and their composition. And also specification of each hardware component can influence the performance of HEVC encoding and decoding process. [1][6]

In section 2, we present basic architecture of HEVC encoder and decoder. In section 3, we describe the basic ideas which we proposed to find proper architecture in multicore platform for HEVC. In section 4, HEVC decoder software is modified to utilize the multicore architecture using parallelizing method and appropriate architecture is proposed based on simulation results. At last section, we define what is done in this paper and finish the paper.

2 Related works

2.1 Feature of HEVC

1) Wavefronts: In the side of increasing the possibility of parallelism in HEVC, we can define the multiple cores can be used in parallel. HEVC adopts the wavefronts concept which splits the picture into CTU rows. This each CTU can be processed in a different processor or thread. So if the architecture has many processing components which can handle the CTU rows, user may configure the encoding or decoding processor can be split into the many CTUs. With multiple processing components, increasing of split depth does not cause increasing of total processing time by using the parallelism. Not only case of wavefronts, there are many possible tasks which can utilize the parallelism of multiple processing components because almost processing unit has similar structure with CTUs.[1][2][6]

2) Slices: HEVC divide a frame into slice which is groups of LCUs in scan order. It can be used for packetization in NAL unit when transmission in network. But more important role of slice is parallel processing in decoding the CUs. Because each thread or processor handle the each slice, so parallelize is improve. But every slice should contain the slice header which includes size, boundaries and parameters. This can distortion transmit rate and the dependencies at each slice’s boundary can interrupt the decoding performance. So, in some cases, exclude slice information when encoding the picture. [1][3][4][6]

3) Dependency: For applying the pipelining or parallelize to HEVC decoder software, Dependency between each tasks or target CUs should be considered as a basis of
smooth decoding process. In case of task’s dependency, all decode tasks should be carried out in regular sequence. So, between neighboring tasks, parallelizing cannot be adopted but pipelining is acceptable. Enter the specific task, for example decompress task, there is dependency between circumjacent CUs because of intra prediction directions as shown in Fig.3. So, when wavefronts is used for parallelizing the decoder, each row of CUs should be decoded after two CUs of previous row are decoded.[1][3][6]

![Wavefronts Processing of CTUs](image1)

Fig. 1. Wavefronts Processing of CTUs

![Slices and Tiles](image2)

Fig. 2. Slices and Tiles

![Dependency Direction in Decompress](image3)

Fig. 3. Dependency Direction in Decompress

Each CU has their address number which start from 0 to (total CU numbers -1). This number cannot be used for slice and tile because they have their numbers. For example, in Fig.2, one tile has 16 CUs so, the address number of each CU is 0 to 15 and other tile has also number of 0 to 15 for included CUs. Foregoing address number can be used for wavefronts parallelize and after-mentioned address number also can be employed for parallelize using allocate each tile to processor or thread. As multi-processor architecture is fastest growing issue in hardware architecture, HEVC concentrates on parallelize decoding process which can make the best use of multi-processor architecture. So, in this paper we modify the HEVC software to put this feature to practical use.

### 2.2 HEVC Decoder

HEVC decoder goes along in the opposite direction of its encoding process. First step of decoder is Read NAL unit task. Because H.264 and HEVC is developed for the purpose of network transmission, encoded data take a form of NAL unit. NAL unit is made for efficiency transmission of picture stream data in network. It consist of parameter set block (SPS, PPS) and slice of picture data. So in this first function identifies what this block is and informs it. Second step is DecodeSlice which consist of initCU, decodeCU and DecompressCU. InitCU just initialize of decoding process. DecompressCU conduct entropy decoding of bitstream. In high efficiency mode of HEVC, entropy encoding conducts based context-based binary arithmetic coding (CABAC). DecompressCU which is the next step of decodeCU in DecodeSlice made up largest number of decoder process. Major roles of this part are dequantization, inverse-transformation, motion compensation and intra/inter prediction. After decompressCU, decoded picture is stored in decoded picture buffer (DPB). This decoded picture is using for deblocking filter (DF), sample adaptive offset (SAO), adaptive loop filter(ALF) and inter prediction. These three filters are part of ExecuteLoopFilter function. So, this step consume much time in access memory. Last step is write picture. When the one picture is decoded completely, this picture is written in decoded picture file. For the most features of each blocks, HEVC have tended to center around the question of how we can achieve more highly resolution pictures and how we can raise a compression ratio with not much more bitstream. But there is no clear solution for architecture to sufficiently handling the most high resolution video stream. To make the best use of HEVC structure, we should put knowledge of characteristics about HEVC to system-level design space exploration. We will see in section 3 how this methodology is being unveiled. [1][4]

![HEVC Decoder Process](image4)

Fig. 4. HEVC Decoder Process

## 3 Proposed methodology of implementing multicore platform for HEVC decoding

In the present paper, we will see the methodology to take appropriate multicore architecture for HEVC decoder.
Before we apply the several CPU architectures, we should analysis HEVC using HM11.0 in the general PC environment so that we choose the primary tasks and define the relation between each task. Some basic features of HEVC can be come to the front from this process. And this process will offer further evidence for the selection of fundamental components of architecture which we will choose as a consideration for simulation. The point which we especially concentrate on is a task which can use the parallelized architecture and consumes much memory bandwidth. Because this is a main feature that different from H.264/MPEG-4. HM11.0 provides the performance information such as bit-size of each slice and total processing time when user performs encoding and decoding process with sample input file. And when we encode the input video file, we can configure most parameters of tasks in HEVC structure. Because as resolution of input video file is increasing and as what the user has more interest between image quality and compression efficiency, composition of encoder options, decoder structure and combination of each parameter set have a broad range. In addition, some tasks do not perform at some cases. So it is material to selecting the main component of architecture by profiling the HEVC using the HM11.0 software as benchmark software.

4 Implementation of HEVC decoder using multicore architecture

According to profiling result of decode function HM11.0, most of performance time is consumed in decompressCU function but initCU and decodeCU functions also have the portion not to be ignored. And because these three functions should be performed in serial with CU order, the dependency between each task is a point to be specially considered. But naturally, decompressCU function is most important part to be parallelize or pipelining for applying to multicore architecture.

HEVC software which is offered from standard association is not optimized to utilize the multicore architecture. And because HEVC is more complicated than H.264 to achieve 50 % higher compression ratio as maintaining the same PSNR, there are some possibility which can draw up the decoding speed using parallelize. But in the multicore environment with OS, it is not easy to guarantee the stable decoding speed.

In this paper, we obtain the trend of decoding speed with various hardware and software architecture using gem5 simulator with changing hardware components and each component’s specification. So in this proposed implementation methodology, both software and hardware can be considered to analyzing decoding speed of HEVC decoder software and proposing appropriate architecture platform.[5]

4.1 Parallelizing of HEVC decoder software

HEVC decoder software is comprised of tiles, slices and CTU rows which are concepts for applying parallelizing. But decoder speed as increasing the number of cores is not changed largely. The reason why decoder speed is not changed is seen plainly through simulation result from gem5 simulator. The stat information which is the result of gem5 simulator tells the fact that however the number of core increased, HEVC decoder software can employ only a one core out of them. So with the original HEVC decoder software, the goal of decoding speed cannot be accomplished. In this reason, openMP is applied to HEVC decoder software. As the profiling results of HM11.0, the most important task is the decode function and CU is the processing unit in that function. Among the ways of using openMP in HEVC decoder software, we choose the decodeCU and decompressCU to parallelize applying region.

We make parallelizing using openMP. In this parallelizing, decodeCU is comprised in parallelizing the decompressCU. Because the time which is consumed during operating a one CU on decodeCU is just one tenth of that of decompressCU, specific CU can be decoded before that CU starts decompress.

But the start CU of each parallelized row should be decoded before it starts. Therefore, address of the number between 1~52 should be decoded before parallelized decompress starts as shown in Fig.6, there are totally three steps for decoding 104 CUs and both second and third steps are applied to openMP. In the decompressCU parallelizing method, each row has a gap of two CUs. It is because of

**Fig. 5. Proposed step-wise methodology**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Profiling using Benchmark software</td>
<td>Modifying the software to utilize a proposed architecture</td>
<td>Defining the concrete specification of each architecture consideration</td>
</tr>
</tbody>
</table>
dependency between CUs when intra-prediction is operated.[3]

![Diagram of HEVC decoder](image)

**4.2 Simulation Results**

We simulate the HEVC decoder software on Gem5 simulator with X86 instruction set and CentOS environment. Default interconnection type is a coherence bus model and multicore type is SMP model. Used decoder software is HM 11.0 and the encoder software is also HM 11.0. HEVC decoder on the gem5 simulator works in practice. Because the Gem5 simulator offers a multicore architecture which has its all components and porting a real OS on the platform, the result of HEVC decoder on simulator is real YUV type video. [5]

First simulation result is decoding speed per number of core. We set the clock speed at 3.3 GHz and memory at 256 MB. Although original HEVC decoder software which we named serial case cannot utilize the multicore architecture, the modified HEVC decoder in section 4 is used to simulate HEVC decoder software on multicore architecture. The region of biggest increasing in decoding speed is between 1 core and 3 cores. And after 3 core, decoding speed over the 30 fps which is the goal of HEVC decoding speed. The max number of threads which is used at the same time is 5 when a one decodeCU row and four decompressCU rows are decoded as shown in Fig.4. So the peak value of decoding speed is achieved when number of cores is 5. After then, decoding speed is decreased when 6 cores are used and it is saturated with 33.3 fps from 7 cores. So, we need more than 3 cores to derive a decoding speed over 30 fps when using multicore architecture with modified HEVC decoder software. And with this modified HEVC decoder software, more cores over 7 is not needed because the decoding speed is saturated after that. And next, same simulation with various clock speeds is done. Under the 3 GHz clock speed, 30 fps is not achieved. With 3.1 GHz clock speed and over the 4 cores, decoding speeds can achieve 30 fps. With all clock speed cases, most significant increasing of decoding speed is shown in range of 1 core to 3 core. But over the 3 cores, there are no regular trends in decoding speed. In some cases, decoding speed is increasing steadily, but in most cases, decoding speeds repeat increasing and decreasing. [2]

Second simulation option is memory size. Tasks which access and use a memory in HEVC decoder software are decompressCU, decode of parameter set, decode of coefficient and three kinds of loop filter. Especially, DPB(Decoded Picture Buffer) occupied much size of memory for saving a temporary decoded picture. Simulation result is represented as decode time of decode function which is composed of decodeCU and decompressCU. Decode speed is stabilized when memory size is over 64 MB. If the memory size is smaller than 30 MB, HEVC decoder stops decoding operation because of resource unavailable. In this simulation environment, OS runs the only a one program. This result can says the minimum memory requirement for decoding HEVC software is 32 MB with 832x480 resolution video file. And to avoid the restriction of memory size, memory size should be over the 64 MB.

Based on these simulation results we can propose minimal cost multicore architecture platform which can satisfy the goal of HEVC decoding speed. Table1 shows that there are many combinations of specification of each component which can achieve the decoding speed of over the 30 fps. So we can choose minimal cost combination set which can make good use of parallelism of HEVC software. The proposed multicore architecture set consists of 3 cores, clock speed of 3.1GHz and memory size of 64 MB with SMP.

| Specification of Multicore Architecture for HEVC |
|---------------------------------|----------------|----------------|
| Clock speed | Memory size | Number of cores |
| Over the 30 fps | 3.1GHz(3 core) ~ 4GHz(1 core) | 52 MB ~ 128 MB | 1core(4GHz) ~8 core |
| Proposed Platform | 3.1 GHz | 64 MB (SMP) | 3 cores (X86) |

5 Conclusion

This paper suggests the multicore architecture which can fully achieve a goal of decoding speed. Target software is HEVC decoder which is next generation video compression coding. So we find the appropriate multicore architecture aimed to specific software using proposed implementing multicore architecture platform methodology. In the first step, we analyze the HEVC decoder software using profiling tool of visual studio 2010 in the first place. The profiling results give information that which task consumes most time in decoding time so that we decide on target task to modify the software for parallelizing. After then we apply the openMP to parallelizing software and proceed the simulation on Gem5 simulator. The main considerations are clock speed, number of cores and memory size. Through simulation, we figure out the final multicore architecture platform which can decode an encoded video file with decoding speed over 30 fps. So we can suggest multicore architecture for HEVC decoder with over the 3.1 GHz clock speed, over the 3 cores and over the 64 MB memory size with modified HEVC software for parallelizing.
6 References


