Multithreading Architecture for Real-Time MPEG-4 AVC/H.264 SVC Decoder

Yong-Hwan Kim, Jiho Park, and Je-Woo Kim

Korea Electronics Technology Institute,
Seongnam-si, Gyeonggi-do, Republic of Korea
{yonghwan,scottie,jwkim}@keti.re.kr

Abstract. The inter-layer prediction (ILP) including intra, residual, and motion up-sampling operation in Scalable Video Coding (SVC) significantly increases the compression ratio compared to simulcast. The SVC Codec capable of processing the inter-layer prediction among multiple layers, however, requires much more memory and computational power than single-layer MPEG-4 AVC/H.264 Codec. This paper presents a fast and memory-efficient multithreading architecture for real-time MPEG-4 AVC/H.264 Scalable High profile decoder. Unlike existing approaches where multi-threaded video encoding and decoding have been performed within a frame or among frames, the designed algorithm utilizes inter-layer parallelism based on a group of macroblocks (GOM). Also, improved buffer management can be achieved by the proposed access unit (AU) based decoding architecture for enabling GOM-based inter-layer multithreading architecture. The proposed multithreading architecture has three properties: (1) scalable to the number of SVC layers, (2) no additional coding delay, and (3) no additional memory requirement. Experimental results show that the proposed multithreading architecture speeds up the decoding time of 3-layer extended spatial scalability sequences by about 36% on average, 3-layer coarse grain scalability 50%, and 5-layer medium grain scalability by about 102%, respectively, compared to a single-threaded SVC decoder.

Keywords: Scalable Video Coding (SVC), Multithreading algorithm, Access unit based decoding.

1 Introduction

Recently video communication services, such as Internet Protocol television (IPTV), mobile IPTV, mobile broadcasting, and multi-screen media service, through various networks and devices have gained growing global interests [1]-[5]. But all these services need to guarantee the minimum quality of service. To meet these service requirements, the new standard which can provide the best quality of service at any environment are required not only in channel coding area but also in source coding area.

To satisfy the increasing industrial needs, ISO/IEC JTC1 SC29 WG11 MEPG and ITU-T SG16 Q.6 VCEG has introduced the new international standard named Scalable Video Coding (SVC) on the end of 2007 as an amendment 3 of MPEG-4

Y.-S. Ho (Ed.): PSIVT 2011, Part II, LNCS 7088, pp. 13–24, 2011.

AVC/H.264 [6]. The SVC can provide a scalable bitstream which supports multiple sub-bitstreams under various spatial, temporal, and quality resolutions [6], [7]. The previous international video coding standards like MPEG-2/H.262 [8], H.263+ [9], and MPEG-4 Visual [10] have already supported various scalable tools. But the previous scalable functionalities of those standards have rarely been used, unfortunately, in the commercial market because of two major reasons [7], [11]. First, the scalable techniques comes along with a significant loss in coding efficiency and a large increase in decoder complexity, compared to those existing alternative solutions. Second, the characteristics of traditional video transmission system were not adequate enough for scalable service.

To overcome drawbacks of the previous scalable tools, the MPEG-4 AVC/H.264 SVC has increased coding efficiency by exploiting inter-layer intra, residual, and motion correlation, also has decreased decoder complexity by introducing single-loop decoding [6], [7]. The complexity of SVC is still high compared to single-layer MPEG-4 AVC/H.264 coding, but additional complexity of SVC mainly arises from considerable memory access for loading and storing data of reference layer (RL) and performing up-sampling operation of inter-layer intra, residual, and motion prediction. It means, if memory-efficient buffer management and complexity reduction on up-sampling process can be achieved, the major limitation of MPEG-4 AVC/H.264 SVC can be solved and thus SVC can be used widely. Especially, fast and memory-efficient up-sampling techniques are essential for real-time Scalable High profile decoder and for low-power SVC decoder on mobile environment. In order to speed up SVC up-sampling operation, Kim, et al. proposed fast and memory-efficient up-sampling methods for extended spatial scalability (ESS), where intra and residual up-sampling operations for ESS are selectively performed by using macroblock (MB) information of enhancement layer (EL) if necessary [12]. Yi, et al. proposed an access unit (AU) based SVC decoding architecture and common residual buffer structure for inter-layer residual prediction (ILRP), in order to significantly reduce memory access and consumption for residual prediction of spatial scalability (SS), coarse grain scalability (CGS), and medium grain scalability (MGS) with and without transform coefficient level prediction [13]. Chunag, et al. analyzed bandwidth overhead of typical SVC decoding and proposed a MB-based up-sampling for spatial scalability and a layer-interleaving decoding scheme for quality scalability in the hardware platform [14]. On the other hand, multi-threaded decoding algorithms for single-layer MPEG-4 AVC/H.264 decoder and various optimization techniques for designing multi-threaded applications have been studied widely [15-21]. The wavefront algorithm can decode several independent MBs concurrently by rearranging the data partition and task scheduling [15]. Chong, et al. presented preparsing technique coupled with run-time MB level scheduling [16]. These methods, however, need considerable number of synchronizations which hurt the overall performance gain obtained from parallel processing [17]. To reduce synchronization overhead, Nishihara et al. proposed a task-parallel approach where a coarse, flexible partitioning adapted for the MPEG-4 AVC/H.264 decoding functions, such as motion compensation, deblocking filtering (DF), and variable length decoding, was developed [18], [19]. Su, et al. proposed a parallel algorithm for SVC decoder, which only supports multi-core stream processor and 3-layer spatial

scalability [21]. Until now, multithreading algorithms for SVC decoder with full function have been rarely studied.

In this paper, AU-based SVC decoding architecture is proposed to implement the proposed multithreading algorithm in the SVC decoder and to reduce memory access and consumption. And, the inter-layer multithreading architecture based on a group of macroblocks (GOM) for real-time Full HD MPEG-4 AVC/H.264 Scalable High profile decoder is presented.

This paper is organized as follows. In section 2 an AU-based SVC decoding architecture is presented. The proposed inter-layer multithreading architecture, and MB-based intra and residual up-sampling methods which are slightly modified version from [12] are presented in section 3. The proposed multithreading algorithm is compared to a single-threaded architecture, in terms of decoding speed, by experiments in section 4, and section 5 concludes the paper.

2 AU-Based SVC Decoding Architecture

An improved AU-based SVC decoding architecture is proposed for considerable reducing memory consumption and bandwidth of ILRP [13]. The proposed method can be applied to both single-threaded (ST) SVC decoder and multi-threaded (MT) one. Note that the proposed algorithm in this section becomes underlying architecture to form the proposed GOM-based inter-layer multithreading architecture.

The SVC supports three types of ILRP, such as transform coefficient level (TCL), transform coefficient (TC), and pixel residual (PR) prediction, for reducing residual signal among layers. Fig. 1 (a) shows one RL MB decoding flow from the viewpoint of ILRP in the JSVM that employs network abstraction layer unit (NALU)-based decoding [22].

In the Fig. 1, the Q^{-1} and IDCT mean inverse quantization and inverse integer discrete cosine transform, respectively. The proposed approach refers to the common residual buffer which stores one of TCL, TC and PR data instead of storing all residual data of RL for ILRP when decoding an EL as shown in the Fig. 1 (b). It is obvious

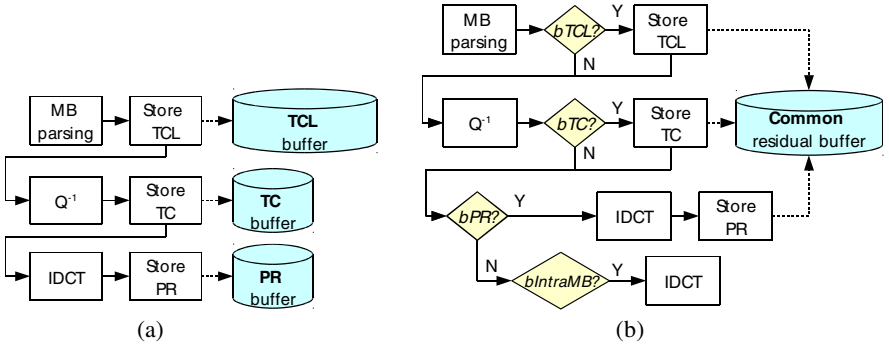

Fig. 1. (a) One RL MB decoding flow from the viewpoint of ILRP operation in the JSVM. (b) One RL MB decoding flow from the viewpoint of ILRP in the proposed AU-based decoding architecture.

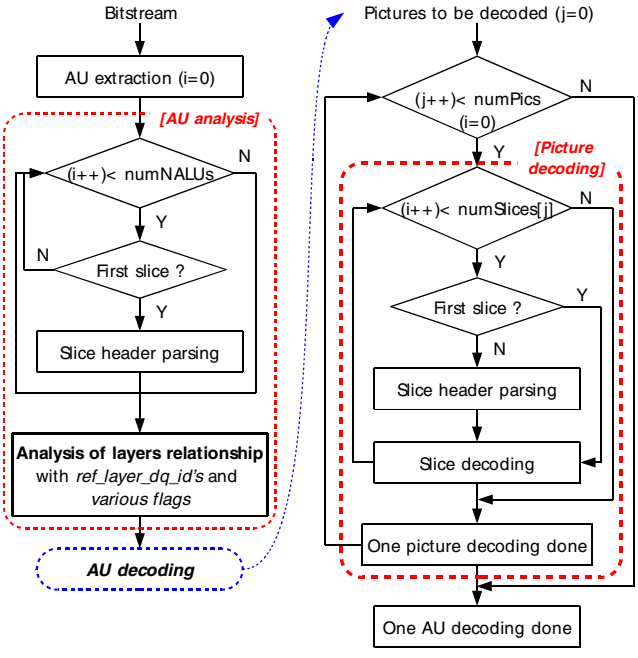

Fig. 2. AU-based SVC decoding architecture

that the proposed method can reduce unnecessarily considerable memory access and consumption in the course of RL decoding. Furthermore, AU-based decoding can eliminate unnecessary IDCT process to store PR data when decoding an RL.

In the Fig. 2, the flowchart of the newly developed AU-based SVC decoding architecture is shown where numNALUs and numPics represent the number of NALUs and pictures within one AU, respectively. The numSlices[j] means the number of slices within a j-th picture. AU-based decoding is performed by two steps: 1) analysis of layers relationship and 2) layers decoding. A SVC AU can be extracted by using the order decision process of NAL unit which is described in subclause 7.4.1.2 and G.7.4.1.2 of the SVC standard [6]. As shown in left part of Fig. 2, picture size, ref_layer_dq_id, and tcoeff_level_prediction_flag values of each layer can be translated by parsing sequence parameter sets and first slice header of each layer before decoding layers. By layer information analysis prior to layers decoding, dependencies among layers can be obtained and what RL data is necessary for ILRP of upper layers can be acquired in advance.

Fig. 3 shows the derivation process of three parameters for fast and memory-efficient ILRP before decoding an RL, where bTCL, bTC, and bPR represent flags whether inter-layer transform coefficient level, transform coefficient, and pixel residual prediction is used or not, respectively. Scalability type of EL can be identified by using picture size information of both EL and the corresponding RL.

As shown in Fig. 3, only one among three parameters should be equal to one. That means storing only one residual data out of three data is sufficient for an RL

decoding. Note that the bTCL, bTC, and bPR parameters can have different values among layers. In the analysis process, it is able to identify layers which are not referred by ELs by comparing all ref_layer_dq_id values and dependency-quality identifiers (DQIDs) of layers prior to AU decoding.

```
If EL is quality layer {
   If tcoeff_level_prefiction_flag of EL == 1 {
     bTCL = 1; bTC = bPR = 0; }
   else {      // tcoeff_level_prediction_flag == 0
      bTC = 1; bTCL = bPR = 0; }
}
else {      // EL is spatial layer
   bPR = 1; bTCL = bTC = 0;
}
```

Fig. 3. Three parameters derivation process for fast and memory-efficient ILRP before decoding a RL picture

As shown in right part of Fig. 2, layers decoding are performed with analyzed layers information. Note that the numPics represents the number of pictures to be decoded. That is, layer pictures which are not referred by ELs are not decoded. The three parameters derived for each layer are input into each layer decoder prior to picture decoding. Fig. 1 (b) shows one RL MB decoding flow from the viewpoint of ILRP in the proposed AU-based decoding architecture. When decoding an RL, only one among TCL, TC, and PR data is stored according to parameters bTCL, bTC, and bPR values. Also, if bPR is not equal to one and MB type is not intra, that is, bIntraMB is equal to zero, IDCT operation is not performed.

The proposed AU-based decoding architecture has four advantages: 1) considerable reduction of memory access and consumption by using only one common residual buffer and selective storing of residual data, 2) reducing computational complexity by effective identifying and skipping pictures not to be decoded, 3) reducing computational complexity by selective skipping unnecessary IDCT operation, and 4) providing easy framework for inter-layer multithreading architecture, which is shown in section 3. In the proposed algorithm, memory access and consumption for ILRP is reduced by half at least, compared to conventional method. That is because the maximum size of the common residual buffer is equal to a size of the existing TCL buffer.

3 GOM-Based Inter-layer Multithreading Architecture for SVC

Based on the algorithms of section 2, the fast and memory-efficient multithreading architecture for real-time Scalable High profile SVC decoder is proposed.

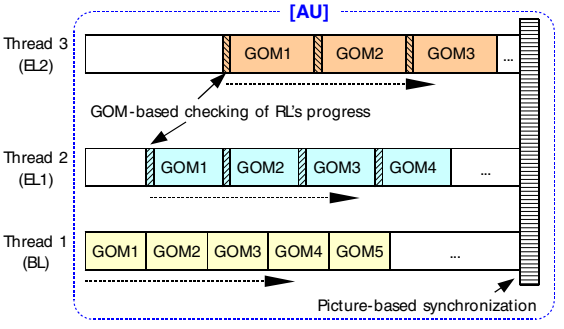

Fig. 4. GOM-based inter-layer multithreading architecture

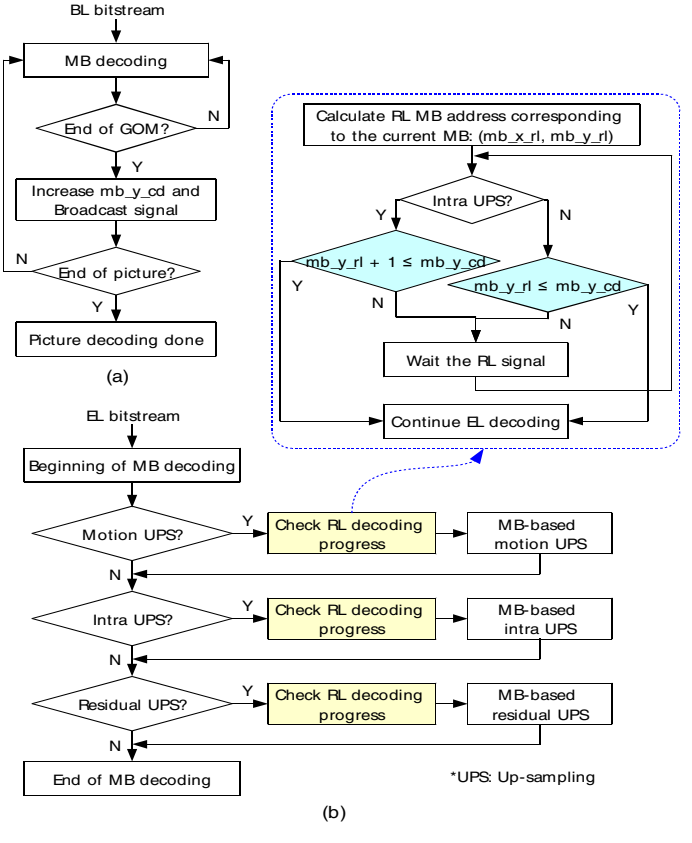

Fig. 5. GOM-based checking of RL decoding progress: (a) BL GOM decoding architecture, (b) EL MB decoding architecture including checking of RL decoding progress

Basically the proposed architecture assigns a thread to one layer decoding. That is, the picture decoding in the right part of Fig. 2 is performed by the corresponding layer threads. An example of the proposed architecture is shown in the Fig. 4, where BL, EL1, and EL2 represent base layer, enhancement layer 1, and enhancement layer 2, respectively. An EL MB can be decoded after decoding the corresponding RL MB since the SVC supports MB-based adaptive inter-layer prediction. Therefore, MB-based synchronization between RL and EL is required for multi-threaded layer decoding, which results in considerable synchronization overhead.

To significantly reduce synchronization overhead, the GOM-based checking of RL's decoding progress as shown in Fig. 4 and Fig. 5 is employed, where the GOM size can be different among layers. In this study, the GOM size of a layer set equal to the number of MBs in a MB row of the layer and thus the GOM size of ELs has the same value as one of BL in the quality scalability. On the contrary, the GOM size of each layer is different in the case of spatial scalability.

Decoding steps are as follows:

(1) BL and RL thread increase mb_y_cd and broadcast completion signal whenever it completes GOM decoding [Fig. 5 (a)].
- The mb_y_cd represents the mb_y address currently decoded in the RL.
- Note that the signal should be broadcasted since one RL can be referenced by one or more EL.

(2) An EL thread checks whether the RL GOM corresponding to the current MB was decoded or not before decoding a GOM of EL [Fig. 5 (b)].
- The checking is performed only if the current MB uses inter-layer prediction.
- If the corresponding GOM of RL is not yet decoded, the EL thread waits the GOM decoding completion signal broadcasted by the RL thread.
- If the corresponding GOM of RL was decoded already, the EL thread continues GOM decoding.

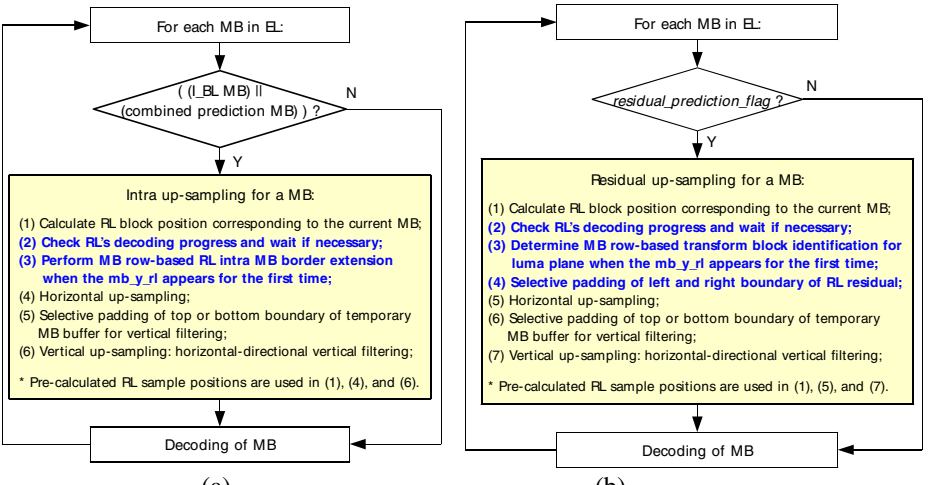

Fig. 6. (a) MB-based intra up-sampling method, (b) MB-based residual up-sampling method

The EL thread waiting overhead is not so high since RL decoding speed is faster than EL decoding speed in most cases. Also, if the EL thread should wait the completion signal in some cases, the waiting is occurred only in the first MB within a GOM of EL. To realize the proposed multithreading architecture, the existing MB-based intra and residual up-sampling methods are modified, which have hybrid architecture combining picture-based pre-processing and MB-based up-sampling [12].

Fig. 6 shows modified MB-based intra and residual up-sampling methods, respectively. In the Fig. 6, the combined prediction MB is a inter MB with two properties that base_mode_flag is equal to 1 and some region of RL block corresponding to the current MB contains intra pixels. The RL decoding progress checking and MB-based up-samplings of Fig. 5 (b) are combined to form complete module in the Fig. 6 (a) and Fig. 6 (b), respectively.

The picture-based pre-processing by EL used in the existing up-sampling methods is not possible in the proposed multithreading architecture since the RL picture is not yet fully decoded prior to EL decoding. To overcome the above problem and to deal with data dependency problem of intra padding process, MB row-based intra MB border extension method is introduced. The proposed intra padding is performed when the mb_y_rl value appears for the first time. Before performing intra padding operation, it should be checked the decoding progress of RL picture since intra MB padding operation requires right and bottom MB types and pixel data. That is, the RL picture should be decoded to (mb_y_rl + 1)-th MB row before padding mb_y_rl-th MB row. Otherwise the EL thread should wait (mb_y_rl + 1)-th MB row decoding completion signal broadcasted by the RL thread as shown in the Fig. 6 (b). On the other hand, determining transform block identification requires that the RL picture should be decoded to mb_y_rl-th MB row before determining that of mb_y_rl-th MB row. Consequently, step (2) and (3) is added to the existing intra up-sampling method in the Fig. 6 (a) and add step (2), (3), and (4) to the existing residual up-sampling algorithm in the Fig. 6 (b).

The proposed multithreading architecture has five properties: 1) applicable to both SVC encoder and decoder, 2) inherently scalable to the number of SVC layers, 3) no additional coding delay, 4) no additional memory requirement, and 5) really much faster than single-threaded SVC architecture, which is proved in following section.

4 Simulation Results

This section covers test conditions and comparison of decoding speed between ST and the proposed MT SVC decoder.

4.1 Test Conditions

To verify the proposed algorithms, a lot of SD (704x576) and Full HD (1920x1080p) sequences are coded with SS, CGS, and MGS configurations by using JSVM encoder version 9.19.8 [22]. Table 1 shows encoding parameters for each test configuration. Experiment was performed four times in a row in the workstation[1]. The purpose of the

[1] Two Zeon X5482@3.2GHz CPU (quad-core), 4 GB DDR2 RAM, and Windows Vista SP2.

first replication was to load the program code and the bitstream into the disk cache and at least partially into the L2 cache [23]. The median value of three other replications was reported. All the experiments are performed by only exchanging MT and ST functions of section 3 in our optimized SVC decoder. The number of threads is the same as the number of SVC layers in the MT decoder experiment.

Note that the algorithms of section 2 and MB-based motion up-sampling method were included in both ST and MT decoder used in the experiment.

Table 1. Encoding parameters of test sequences

Parameters	Value	
Profile	Scalable High	
Frame rate [Hz]	Full HD (25), SD (30)	
# frames	Full HD Bluesky (217), Other Full HDs (300), SD (200)	
Intra period	32	
# reference frame	1	
ME mode	Fast log-search (Search mode: 4)	
Search range	Full HD (96), SD (64)	
Loop filter	On (0)	
Tools	8x8 trans., Adaptive inter-layer intra/motion/residual prediction	
Configuration A (3 layers SS)	720x480p/1280x720p/1920x1080p Entropy: CAVLC GOP: 8 (Hierarchical-B)	: 3-layer ESS with wide range of quality, which is a typical scenario for multi-screen media service
Configuration B (3 layers Full HD CGS)	1920x1080p Entropy: CABAC GOP: 4 (Hierarchical-B)	: High quality CGS with CABAC and low QP, which represents a seamless Full HD movie streaming service scenario in the best-effort network
Configuration C (3 layers SD CGS)	704x576 Entropy: CAVLC GOP: 8 (Hierarchical-B) tcoeff_level_prediction_flag = 1	: Medium quality CGS with MPEG-4/H.264 rewriting capability, which represents a backward-compatible SD movie streaming service scenario
Configuration D (5 layers Full HD MGS)	1920x1080p Entropy: CAVLC GOP: 4 (Hierarchical-B) Scan_idx:[0,15]/[0,2]/[3,6]/[7,10]/[11,15]	: 5-layer MGS with wide range of quality, which represents more accurate quality control scenario over CGS

4.2 Decoding Speed

Table 2 shows the decoding speed comparison between ST and the proposed MT SVC decoder with test configuration A, where fps and kbps represent frame per second and kilobits per second, respectively. In this scenario, MT decoder is faster than ST decoder by about 36.5% on average. The proposed MT decoder shows real-time decoding for all sequences and all QPs except one high quality (QP=18) Tractor sequence which has high texture background and camera motion. In fact, the high quality Tractor sequence is not adequate for consumer video service since the bitrate is too high, i.e., 46 Mb/s, as shown in the Table 2.

Table 3 and 4 show the comparison results with test configuration B and C, respectively. The MT decoder is faster than ST decoder by about 50.4% and 72.1% on average, respectively, in these scenarios. The decoding speed of test configuration B is relatively slower than the other configurations due to CABAC entropy decoding.

Table 2. The decoding speed comparison between a ST and the proposed MT decoder for test configuration A

Parameters	QP	PSNR[dB]	Bitrate[kbps]	ST[fps]	MT[fps]	Gain[%]
Bluesky (BS)	18	44.00	39352.41	19.03	25.28	32.84
	23	41.63	16643.46	27.00	36.02	33.41
	28	39.75	7996.60	34.20	46.49	35.94
	33	37.95	4773.65	36.62	50.20	37.08
Pedestrian (PE)	18	44.43	26020.70	21.95	29.42	34.03
	23	42.96	10548.51	28.99	40.22	38.74
	28	41.62	4670.44	34.04	48.01	41.04
	33	40.22	3078.02	36.62	52.22	42.60
Rushhour (RH)	18	43.61	25763.03	19.50	25.83	32.46
	23	43.02	9451.03	26.98	36.93	36.88
	28	42.00	3769.93	33.78	46.78	38.48
	33	41.02	2561.92	37.20	51.74	39.09
Sunflower (SF)	18	44.78	18620.32	25.95	34.57	33.22
	23	43.49	6903.74	35.86	49.39	37.73
	28	42.03	3111.94	41.73	57.79	38.49
	33	40.31	1944.26	43.93	61.32	39.59
Station (ST)	18	43.43	25718.22	24.22	30.67	26.63
	23	41.95	9061.29	37.80	52.28	38.31
	28	40.85	3156.49	43.12	61.67	43.02
	33	39.34	1826.15	45.45	64.71	42.38
Tractor (TR)	18	43.32	46305.29	15.39	20.11	30.67
	23	41.04	20056.58	21.22	28.07	32.28
	28	39.30	9456.63	26.53	35.80	34.94
	33	37.89	5877.10	29.12	39.36	35.16
Average	25.5	41.66	12777.82	31.09	42.70	36.46

Table 3. The decoding speed comparison between a ST and the proposed MT decoder for test configuration B

Seq.	QP	PSNR[dB]	Bitrate[kbps]	ST[fps]	MT[fps]	Gain[%]
BS	29/24/20	43.49	25790.61	19.70	29.89	51.73
PE	28/24/18	44.84	28386.58	18.95	27.82	46.81
RH	28/23/18	44.78	28113.36	18.28	26.53	45.13
TR	28/22/18	43.45	52549.38	12.82	20.22	57.72
Avg.	28/23/19	44.14	33709.98	17.44	26.12	50.35

Table 5 shows the comparison results for test configuration D. The proposed MT decoder is faster than ST decoder by about 101.7 % on average. In this MGS scenario, the ST decoder cannot decode all bitstreams in real-time. The MT decoder, however, shows real-time decoding for all sequences and all QPs.

Speed-up gains for four test configurations is quite different due to variable thread synchronization overhead and load imbalance between layers. Especially, the proposed multithreading architecture shows higher speed-up gain for CGS and MGS decoding since inherently quality scalability has more balanced load between layers than spatial scalability.

Table 4. The decoding speed comparison between a ST and the proposed MT decoder for test configuration C

Seq.	QP	PSNR[dB]	Bitrate[kbps]	ST[fps]	MT[fps]	Gain[%]
City	33/28/23	40.70	5636.54	123.16	207.05	68.11
Harbour	33/28/23	40.62	9740.48	93.15	144.85	55.50
Ice	33/28/23	44.73	2552.87	161.26	312.72	93.92
Soccer	33/28/23	41.55	5693.56	121.95	208.13	70.67
Avg.	33/28/23	44.14	5905.86	124.88	218.19	72.05

Table 5. The decoding speed comparison between a ST and the proposed MT decoder for test configuration D

Seq.	QP	PSNR[dB]	Bitrate[kbps]	ST[fps]	MT[fps]	Gain[%]
BS	24/18	44.79	44891.22	15.73	33.60	113.60
	29/23	42.24	20704.44	19.49	38.85	99.33
	34/28	40.13	10984.17	22.45	43.73	94.79
PE	24/18	45.07	34133.67	16.91	36.06	113.25
	29/23	43.24	12905.67	21.59	42.90	98.70
	34/28	41.76	6591.23	24.02	46.50	93.59
RH	24/18	44.82	35398.54	16.29	34.26	110.31
	29/23	43.33	11756.15	21.45	41.90	95.34
	34/28	42.17	5471.36	24.49	46.60	90.28
TR	24/18	44.14	54517.41	14.09	30.88	119.16
	29/23	41.64	24451.88	17.94	36.08	101.11
	34/28	39.69	12414.59	21.01	40.09	90.81
Avg.	29/23	42.75	22851.69	19.62	39.29	101.69

5 Conclusion

In this paper the proposed fast and memory-efficient multithreading architecture and AU-based SVC decoding architecture was described. The proposed AU-based SVC decoding architecture reduced memory access and consumption by half at least by analyzing relationship of layers. The proposed inter-layer multithreading architecture speeds up the SVC decoder up to 101% without additional coding delay and memory consumption, compared to a single-threaded decoder.

Although the implementation of the proposed multithreading architecture was done only in the SVC decoder with SS, CGS, and MGS, the proposed architecture can also be applied to SVC encoder. Furthermore, the multithreading architecture can be implemented on any general-purpose processor, such as x86 and ARM.

References

1. Wiegand, T., Noblet, L., Rovati, F.: Scalable Video Coding for IPTV services. IEEE Trans. Broadcasting 55(2), 527–538 (2009)
2. Schierl, T., Stockhammer, T., Wiegand, T.: Mobile video transmission using Scalable Video Coding. IEEE Trans. Circuits Syst. Video Technol. 17(9), 1204–1217 (2007)

3. ATSC Mobile DTV Standard, Part 7 - AVC and SVC Video System Characteristics: A/153 Part7:2009, ATSC (2009)
4. Choi, H., Shin, I.H., Lim, J.-S., Hong, J.W.: SVC application in advanced T-DMB. IEEE Trans. Broadcasting 55(1), 51–61 (2009)
5. Tan, P., Slevinsky, J.: Multi-screen IPTV enabling technologies and challenges. In: Proc. IEEE Int. Conf. Consumer Electronics, pp. 1–2 (2011)
6. Advanced video coding for generic audiovisual services: ITU-T Rec. H.264 and ISO/IEC 14496-10, ITU-T and ISO/IEC JTC 1 (2010)
7. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the Scalable Video Coding extension of the H.264/AVC standard. IEEE Trans. Circuits Syst. Video Technol. 17(9), 1103–1120 (2007)
8. Generic Coding of Moving Pictures and Associated Audio Information Video: ITU-T Rec. H.262 and ISO/IEC 13818-2, ITU-T and ISO/IEC JTC 1 (2000)
9. Video Coding for Low Bit Rate Communication: ITU-T Rec. H.263, ITU-T (2005)
10. Coding of Audio-Visual Objects - Part 2 Visual: ISO/IEC 14496-2, ISO/IEC JTC 1 (2004)
11. Liu, H., Wang, Y.-K., Li, H.: A comparison between SVC and transcoding. IEEE Trans. Consumer Electronics 54(3), 1439–1446 (2008)
12. Kim, Y.-H., Yi, J.-Y., Choi, B.: Fast and memory-efficient up-sampling methods for H.264/AVC SVC with extended spatial scalability. IEEE Trans. Consumer Electronics 56(2) (2010)
13. Yi, J.-Y., Kim, Y.-H., Choi, B.: Fast and memory-efficient AU-based decoding method for H.264/AVC SVC. In: Proc. CEWIT, Incheon, Korea (2010)
14. Chuang, T.-D., Tsung, P.-K., Lin, P.-C., Chang, L.-M., Ma, T.-C., Chen, Y.-H., Chen, L.-G.: Low bandwidth decoder framework for H.264/AVC Scalable Extension. In: Proc. IEEE Int. Symp. Circuits and Systems, pp. 2960–2963 (2010)
15. van der Tol, E.B., Jaspers, E.G.T., Gelderblom, R.H.: Mapping of H.264 decoding on a multiprocessor architecture. In: Proc. SPIE Conf. on Image and Video Communications and Processing, pp. 707–718 (2003)
16. Chong, J., Satish, N., Catanzaro, B., Ravindran, K., Keutzer, K.: Efficient parallelization of H.264 decoding with macro block level scheduling. In: Proc. IEEE ICME, pp. 1874–1877 (2007)
17. Yang, S.-S., Wang, S.-W., Wu, J.-L.: A parallel algorithm for H.264/AVC deblocking filter based on limited error propagation effect. In: Proc. IEEE ICME, pp. 1858–1861 (2007)
18. Nishihara, K., Hatabu, A., Moriyoshi, T.: Parallelization of H.264 video decoder for embedded multicore processor. In: Proc. IEEE ICME, pp. 329–332 (2008)
19. Sihn, K.-H., Baik, H., Kim, J.-T., Bae, S., Song, H.J.: Novel approaches to parallel H.264 decoder on symmetric multicore systems. In: Proc. IEEE ICASSP, pp. 2017–2020 (2009)
20. Kim, D., Lee, V.W., Chen, Y.-K.: Image processing on multicore x86 architecture. IEEE Signal Processing Magazine 27(2), 97–107 (2010)
21. Su, Y.-C., Tsai, S.-F., Chuang, T.-D., Tsao, Y.-M., Chen, L.-G.: Mapping Scalable Video Coding decoder on multi-core stream processors. In: Proc. Picture Coding Symposium, pp. 1–4 (2009)
22. Reichel, J., Schwarz, H., Wien, M.: Joint Scalable Video Model 11 (JSVM 11): JVT, Geneva, CH, Doc. JVT-X202 (2007)
23. Lappalainen, V., Hallapuro, A., Hamalainen, T.D.: Complexity of optimized H.26L video decoder implementation. IEEE Trans. CSVT 13(7), 717–725 (2003)