# Memory Analysis for H.264/AVC Scalable Extension Decoder

Po-Yuan Hsu, Gwo-Long Li, and Tian-Sheuan Chang
Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan
E-mail: pyhsu.ee96g@nctu.edu.tw, glli.ee95g@nctu.edu.tw, tschang@twins.ee.nctu.edu.tw

***Abstract*— In this paper, a systematic analysis for memory usage in H.264/AVC scalable extension (SVC) decoder is presented. This paper analyzes the memory requirements with three different decoding flows, macroblock, row and frame based, to find out a best method which can achieve optimal trade-off between internal memory usage and external memory access. The analysis shows that the SVC decoding needs 88% to 110% extra memory bandwidth compared to single layer H.264 decoding due to inter-layer prediction disregarding of decoding flow. However, extra internal memory storage by inter-layer prediction varies a lot according to the flow. This analysis could provide as a foundation to design a SVC decoder for further step.**

***Index Terms*—** scalable video coding (SVC), decoder

## I. Introduction

Recently, the advances of network bandwidth and wireless access techniques boost the development of multimedia services. The state-of-the-art video codec H.264/AVC promises the dominant status over multimedia content service. It provides high compression and high quality video but with only fixed resolution. Due to the heterogeneities on user devices and network environments, multimedia stream with scalable features is demanded. A single bitstream to satisfy various clients becomes more and more desired. Therefore, Scalable Video Coding (SVC) [1] is introduced to provide this service.

SVC is a new video codec based on H.264/AVC, currently being normalized by the Joint Video Team (JVT). It provides multiple display resolution, frame rate, and video quality within a single bitstream. SVC allows three scalabilities (spatial, temporal, and quality scalability) [1]. Figure 1 shows the architecture of SVC encoder with two spatial layers. Spatial scalability consists of three new inter-layer prediction modes. When these modes are chosen, the corresponding data is upsampled from previous layer.

Due to the spatial scalability, memory requirement of SVC decoder is different from previous H.264 decoder [3]. The main difference is that the decoded data need to be stored for inter-layer prediction. Therefore, the memory requirements are different between decoding flows if we taking the inter-layer prediction into consideration.

In order to reuse inter-layer data efficiently, this paper analyzes the internal memory and external memory requirements for three different decoding flows, macroblock-based, row-based, and frame-based methods. By the assistant of this analysis results, a better way for SVC decoder hardware design can be found.

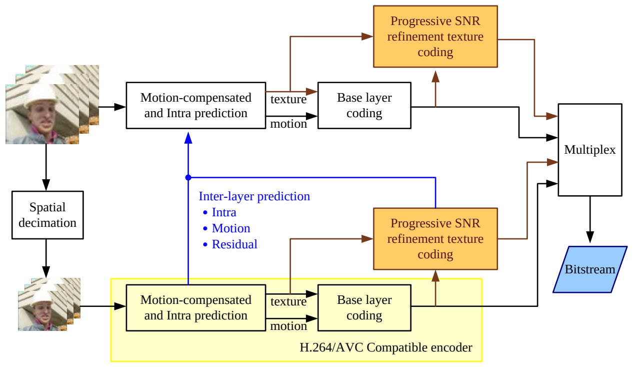
Progressive SNR refinement texture coding
Motion-compensated and Intra prediction
texture
motion
Base layer coding
Multiplex
Spatial decimation
Inter-layer prediction
• Intra
• Motion
• Residual
Progressive SNR refinement texture coding
Bitstream
Motion-compensated and Intra prediction
texture
motion
Base layer coding
H.264/AVC Compatible encoder

Fig. 1 Architecture of SVC encoder with two spatial layers.

The organization of this paper is described as follows. Section II shows the overview of H.264 and SVC decoder. The memory requirements of H.264 and three different SVC decoding flows are described in Section III. Section IV exhibits the analysis results among different situations. Then we make conclusions in Section V.

## II. Overview of H.264/AVC and SVC Decoder

For an H.264/AVC decoder, a straightforward decoding process is macroblock by macroblock. The decoding flow first parses the input bitstream by entropy decoding and recovers the residual and related prediction mode data needed for the decoding process. Then the residual is added with the prediction samples from inter or intra prediction to recover the pixel values.

In addition to the inherent decoding operations of H.264, the SVC decoder supports three scalabilities, spatial, temporal, and quality. For the purpose of memory analysis, we only consider spatial scalability in this paper since temporal scalability should be fully supported. For the spatial scalability, base layer data needs to be decoded before its corresponding macroblock in enhancement layer for the purpose of inter-layer data reuse. Enhancement layer reuses base layer information such as motion vectors, residuals, or reconstructed samples for the reconstruction process. The block diagram of SVC decoder is shown in Figure 2. There are three new blocks in the figure within dashed line compared to H.264. In which motion vector upsampling upsamples macroblock partition and motion vectors from

The work is supported by National Science Council of Taiwan, under Grant NSC 97-2220-E-009-018.

reference layer. Residual upsampling is corresponding to inter-layer residual prediction and it block-wised upsamples residuals. Sample upsampling process is used in inter-layer intra texture prediction that upsamples the corresponding reconstructed samples. With these inter-layer predictions, SVC decoder can decode its data to recover the pixel values.

## III. Memory Analysis

### A. *Memory Analysis for H.264/AVC Decoder*

The memory requirements of H.264 decoder are mainly dominated by four parts: parsing data, macroblock processing data, neighboring data, and decoded picture buffer. In the following, a macroblock based decoding flow is assume. The parsing data stores the information parsed from the bitstream, such as the SPS, PPS, and slice header data. The transform coefficients parsed from bitstream are also included in it. The parsing data consumes about 4.4 KB memory derived statistical results. The macroblock processing data stores the information that may be used to reconstruct a macroblock such as prediction mode, residuals, and reconstructed samples. This part requires 4.8 KB memory spaces. Since the basic processing unit is macroblock, above memory usage is irrelevant to the frame size actually. The neighboring data stores previous decoded neighboring macroblock information, i.e. left, up, upper-right, or upper-left macroblock, which contains motion vectors, reference picture, prediction mode, and neighboring pixels. Pre-deblocking coefficients are also included. The neighboring data is stored in a row of macroblocks in frame width, for example, a row consists of 22 macroblocks in CIF size. For neighboring pixels, the size is one line of samples of the frame width plus one column height of a macroblock. The decoded picture buffer (DPB) stores previous decoded frames as reference frame for inter prediction. The DPB is refreshed after decoding one GOP. However, the data of DPB are stored in external memory since such significant memory requirement is unreasonable to be stored in internal memory. From the analysis described above, the internal memory usage and external memory access of H.264 decoder are summarized in Table I and Table II, respectively.

### B. *Memory Analysis for SVC Decoder*

Memory requirement of SVC inherits the entire requirement from H.264 with additional memory from inter-layer prediction as shown in Figure 3. For the inter-layer prediction, it reuses the reference layer data such as motion vectors, residuals, and reconstructed samples. These data are recognized as inter-layer data. With this inter-layer dependency, different decoding flows will cause different memory requirements. In this paper, we specify three decoding flows, macroblock-based, row-based, and frame-based, that can be applied to SVC decoding process. Furthermore, their corresponding internal memory usage and external memory access will be analyzed in the following article.

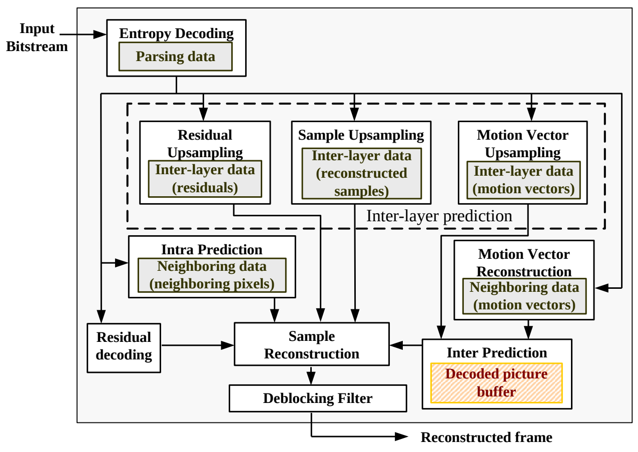

Fig. 2 Block diagram of SVC decoder.

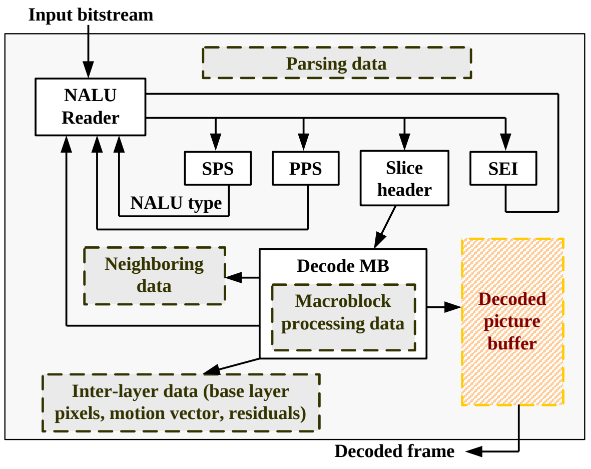

Fig. 3 SVC decoder memory map

TABLE I. Internal memory usage of H.264/AVC decoder

| Name | Size (KB) |
|---|---|
| *Parsing data | ~4.4 |
| *Macroblock processing data | ~4.8 |
| Neighboring data | 0.2 * PicWidthInMbs + 1.06 |

*PicWidthInMbs: number of macroblocks in a row of a frame*

**: fixed data: will not change with the variation of frame resolution*

TABLE II. External memory access of H.264/AVC decoder

| Name | Size (KB) |
|---|---|
| Input bitstream | *0.0375*PicSizeInMbs |
| Reference samples | **1.35 *PicSizeInMbs |
| Reconstructed samples | 0.375*PicSizeInMbs |

*PicSizeInMbs: number of macroblocks in a frame*

**: assume the compression rate is 10% of the original data*

***: assume every macroblock has 16 4x4 subblocks with one direction prediction*

### *1) Macroblock (MB)-based*

The first decoding flow is macroblock-based, which decodes one macroblock in base layer and then the corresponding four macroblocks in the enhancement layer. In this flow, inter-layer data can be reused immediately for enhancement layer. Figure 4(a) illustrates the pseudo code of this decoding method. In this method, we have to store the neighboring data corresponding to its current decoding macroblock of each spatial layer separately since multiple layers are decoded at the same time. Therefore, the internal memory usage, especially the neighboring data, is increased with more layers as shown in Table III.Besides, we also have to store the base layer decoded information as inter-layer data for inter-layer prediction reference. Therefore, Table IV shows the composed elements of inter-layer data and its corresponding additional memory requirement. The $\text{NumMB}_{\text{ref}}$ is set as follows:

$$\text{NumMB}_{ref} = \sum_{n=0}^{d-2} 2^{2n} . \qquad (1)$$

*d: number of spatial layers*

In this method, these data can be immediately reused in the following enhancement layer decoding. Table V shows the corresponding external memory access.

### *2) Row-based*

Row based decoding expands the decoding flow to row by rows. After decoding one row of macroblocks in base layer, the corresponding two rows in higher enhancement layer will be decoded as shown in Figure 4(b). In this flow, inter-layer data is reused for enhancement layer after one row processing. Row-based decoding flow is similar to macroblock-based decoding flow since both methods decoding multiple layers at the same time. However, the control of row-based decoding method is much easier than macroblock-based method since the decoding process change between each layer is more regular, i.e. the decoding macroblocks are continuous in each layer. The main difference of memory requirement between row-based and macroblock-based is that the size of inter-layer data. The row-based manner should buffer whole row(s) of inter-layer data. The $\text{NumMB}_{ref}$ in Table IV in this method is set as follows:

$$\text{NumMB}_{ref} = \sum_{n=0}^{d-2} \text{PicWidthInMbs}_n . \qquad (2)$$

*d: number of spatial layers*

*PicWidthInMbs$_n$: number of macroblocks in a row in spatial layer n*

The external memory access is almost the same as macroblock-based method.

### *3) Frame-based*

Frame-based decoding means that the decoder processes each frame layer by layer. In this flow, inter-layer data is reused for enhancement layer after one frame processing. The pseudo code is shown in Figure 4(c). The frames of enhancement layer are decoded after the base layer decoding. This method

TABLE III. INTERNAL MEMORY USAGE OF MULTI-LAYER DECODING

| Name | Size (KB) |
|---|---|
| Parsing data | ~4.5 |
| Macroblock processing data | ~5 |
| Neighboring data | $\sum_{n=0}^{d-1} (0.2 * \text{PicWidthInMbs}_n + 1.06)$ |

*d: the number of spatial layers*

*PicWidthInMbs$_n$: number of macroblocks in a row of spatial layer n*

TABLE IV. INTER-LAYER DATA REQUIREMENT OF SVC

| Name | Size (KB) |
|---|---|
| Reference layer motion vectors | $0.125 * \text{NumMB}_{ref}$ |
| Reference layer residuals | $0.75 * \text{NumMB}_{ref}$ |
| Reference layer samples | $0.375 * \text{NumMB}_{ref}$ |
| Reference layer others (mb_type, sub_mb_type, ref_idx, …) | $0.013 * \text{NumMB}_{ref}$ |

*NumMB$_{ref}$: number of macroblocks of reference layer*

TABLE V. EXTERNAL MEMORY ACCESS OF SVC DECODER

| Name | Size (KB) |
|---|---|
| Input bitstream | $0.0375 * \sum_{n=0}^{d-1} \text{PicSizeInMbs}_n$ |
| Reference samples | $1.35 * \sum_{n=0}^{d-1} \text{PicSizeInMbs}_n$ |
| Reconstructed samples | $0.375 * \sum_{n=0}^{d-1} \text{PicSizeInMbs}_n$ |
| FGS coefficient | $0.75 * q * \sum_{n=0}^{d-1} \text{PicSizeInMbs}_n$ |

*d: number of spatial layers*

*PicSizeInMbs$_n$: number of macroblocks in a frame*

*q: number of FGS layers*

(a)
```
while( spatial_layer_id < d ){
  while( CurrMbAddr < $4^{spatial\_layer\_id}$ ){
    decode_one_macroblock();
    CurrMbAddr ++;
  }
  spatial_layer_id ++;
}
```

(c)
```
while( spatial_layer_id < d ){
  while( CurrMbAddr < PicSizeInMbs ){
    decode_one_macroblock();
    CurrMbAddr ++;
  }
  spatial_layer_id ++;
}
```

(b)
```
while( spatial_layer_id < d ){
  while( CurrMbAddr < PicWidthInMbs * $2^{spatial\_layer\_id}$ ){
    decode_one_macroblock();
    CurrMbAddr ++;
  }
  spatial_layer_id ++;
}
```

*spatial_layer_id: the index of current decoding spatial layer*

*d: number of spatial layers*

*CurrMbAddr: current decoding macroblock address*

*PicWidthInMbs: picture width in the unit of MBs*

*PicSizeInMbs: picture size in the unit of MBs*

Fig. 4 Pseudo code for three decoding flows: (a) macroblock-based, (b) row-based, (c) frame-based

has to store a whole frame of inter-layer data. The amount of inter-layer data is to substitute (3) into Table IV.

$$\text{NumMB}_{ref} = \sum_{n=0}^{d-2} \text{PicSizeInMbs}_n \ . \tag{3}$$

*d: number of spatial layers*
*PicSizeInMbs$_n$: number of macroblocks in spatial layer n*

However, these huge inter-layer data are unreasonable to be stored in internal memory. Furthermore, due to different layers are decoded at different time, neighboring data can be stored in only one set of highest spatial layer without overlap. Therefore, the internal memory size is reduced to by setting d = 1 in Table III when compared to macroblock-based and row-based decoding manners. The total external memory access is same as Table V plus inter-layer data mentioned above.

## IV. Results and Discussions

For a clearer picture of the memory usage, we show some quantitative results in this Section. To calculate the memory usage, we make several assumptions in our analysis: 95% of macroblocks are coded in inter prediction, 90% of macroblocks in enhancement layers are coded in Intra_BL mode if the corresponding block in base layer is encoded as Intra mode, and 10% macroblocks in enhancement layers use residual prediction in average. This assumption is a general statistic according to our experiment. The encoding settings are listed in Table VI. Table VII shows the analysis results, in which type I stores all inter-layer prediction data in the internal storage while type 2 stores all inter-layer prediction data in the external memory.

The result shows that inter-layer prediction data has great impact to both internal memory storage as well as the external memory access. For type I, internal memory size will be increased by 81% to 8965% when compared to single layer H.264 decoding, especially for frame-based decoding that needs to store 1980 MBs of residuals, prediction modes and motion vectors for inter-layer prediction. For row-based decoding method, 110 MBs of inter-layer prediction data need to be stored in internal memory. For MB-based decoding, only 5 MBs data are needed. This is the reason why the macroblock-based decoding method results in lower internal memory usage. For external memory access, all these flows are the same due to the same reference data. Thus, the MB-based decoding is the best choice due to its smallest internal memory usage and the same external memory access when compared to frame-based and row-based decoding methods, if an efficient internal memory design can be supported by the technology provider.

Beyond type I, another design possibility is to store the inter-layer prediction data into external memory to reduce the chip cost, just as type II. From the table, it is interesting to find that the extra external memory bandwidth due to inter-layer prediction data is insignificant compared to the large reference data. Thus, the bandwidth increasing in type II is just 12% more when compared to that in type I. However, the internal memory usage varies a lot for different coding flow. MB-based decoding has the least reduction due to each layer has its own neighboring data to be stored in internal memory for each layer decoding. Same situation also occurs in row-based decoding method as well. For frame-based decoding, the internal memory storage is just 1% more than the single layer H.264 decoding since all the extra storage is within the external memory now. Therefore, the frame-based decoding is the best choice for smallest internal memory size with the acceptable memory bandwidth.

Table VI Simulation Settings

| | |
|---|---|
| GOP | 8 |
| QP | 32, 26, 20 |
| Intra period | -1 |
| Frame resolution | QCIF, CIF, 4CIF |

TABLE VI. Table VII comparison of Memory requirements

| Decoding Flow | | Internal Memory (KB) | | External Memory Access (MB) | |
|---|---|---|---|---|---|
| | | size | ratio | size | ratio |
| Original H.264 | | 27.9 | 100% | 11.5 | 100% |
| Type I | MB | 50.6 | 181% | 21.7 | 189% |
| | Row | 126.6 | 454% | 21.5 | 187% |
| | Frame | 2529 | 9065% | 21.4 | 186% |
| Type II | MB | 44.3 | 159% | 24.2 | 210% |
| | Row | 43.5 | 156% | 24 | 209% |
| | Frame | 28.2 | 101% | 23.9 | 208% |

## V. Conclusions

In this paper, the memory requirement of SVC decoder is analyzed for three SVC decoding flows. Analysis results show that MB-based or row-based decoding can reuse the inter-layer data but needs extra storage. These two can be a design choice if efficient on-chip memory technology can support. For lower internal memory size and low external memory access, frame-based decoding with external storage of inter-layer data can be a better choice for SVC decoder design.

## References

[1] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Tran. CSVT*, vol. 17, no. 9, pp.1103-1120, Sep. 2006.

[2] H. Schwarz, D. Marpe, and T. Wiegand, “Hierarchical B Pictures,” Joint Video Team, Doc. JVT-P014, Jul. 2005.

[3] M. Pelcat, M. Blestel, M. Raulet, “From AVC decoder to SVC: minor impact on a dataflow graph description,” *in Proc. IEEE PCS*, Lisboa, Portugal, Nov. 2007.