AN H.264/SVC MEMORY ARCHITECTURE SUPPORTING SPATIAL AND COURSE-GRAINED QUALITY SCALABILITIES

Niranjan D. Narvekar, Bharatan Konnanath, Shalin Mehta, Santosh Chintalapati, Ismail AlKamal, Chaitali Chakrabarti and Lina J. Karam

Department of Electrical Engineering,
Arizona State University, Tempe, AZ 85287-5706, USA

ABSTRACT

The standardized Scalable Video Coding (SVC) extension of H.264/AVC achieves significant improvements in coding efficiency relative to the scalable profiles of prior video coding standards, but its computational complexity and memory access requirements make the design of a low power hardware architecture a challenging task. This paper presents an SVC decoder architecture supporting spatial and coarse-grained quality scalability. The architecture optimizes the size of the on-chip memory and reduces the power-consuming and time-intensive external memory accesses.

Index Terms— H.264/AVC, Scalable Video Coding (SVC), CGS, decoder architecture

1. INTRODUCTION

The Scalable Video Coding (SVC) extension of the video coding standard H.264/AVC [1, 2, 3, 4] promises superior coding efficiency over the scalable profiles of previous video coding standards including H.262/MPEG-2, H.263, and MPEG-4. The scalability of video enables easy adaptation to device capabilities and channel conditions, higher transmission robustness and ease of multicast streaming over heterogeneous networks without the necessity of transcoding.

Several architectures [5, 6, 7, 8] have been proposed for efficient H.264/AVC decoder implementations that meet real-time constraints. ASIC architectures have also been proposed for optimizing specific blocks of the decoder, such as the in-loop deblocking filter [9] which consumes one-third of the total CPU time. H.264 SVC inherits the complexity of H.264/AVC. In addition, the presence of multiple layers of video data further increases the demand on the on-chip memory. In order to enable the adoption of H.264 SVC in current and emerging low-power multimedia devices, it is necessary to design low-power and high throughput hardware implementations supporting real-time encoders and decoders.

This paper presents an ASIC H.264/SVC decoder architecture that supports spatial scalability and coarse-grained quality scalability (CGS). The main focus of this work is on designing an efficient memory management scheme which minimizes the number of memory accesses from the external memory while limiting the size of the required on-chip memory. The proposed architecture makes use of a scratchpad memory as the on-chip memory to reduce power consumption. It builds on existing blocks of H.264/AVC while adding on the necessary blocks to support SVC spatial scalability and CGS.

The rest of the paper is organized as follows. Section 2 provides a brief overview of H.264 SVC. The proposed architecture is described in Section 3 which is followed by a dataflow and memory requirement analysis in Section 4. The conclusions are given in Section 5.

2. BACKGROUND

The SVC extension of H.264/AVC supports temporal, spatial and quality scalability. Spatial scalability follows the conventional approach of multi layer coding. In each spatial layer, motion compensated predictions and intra predictions are employed as in single layer coding. H.264 SVC includes additional inter-layer prediction mechanisms, which result in improved coding efficiency. Two types of quality scalabilities are supported namely, coarse-grained scalability (CGS), which supports bit rate adaptation at the level of coded video sequences, and medium-grained scalability (MGS), which supports bit rate adaptation at a NAL unit level [2].

The decision process at the decoder for decoding the enhancement layer macroblocks (MBs) is shown in Fig. 1. The type of enhancement layer macroblock (MB) is signaled using a syntax element called *base mode flag* (BMF). In addition to the inter-picture motion prediction and intra-picture prediction modes as in H.264/AVC, the H.264 SVC codec supports various inter-layer prediction modes (Fig. 1) including:

1) *Inter-Layer Intra Prediction:* When BMF is 1 and the co-located reference layer block is intra coded, then inter-layer intra prediction is used. In this mode, the co-located reference layer intra coded block is up-sampled and added to the enhancement layer residual to obtain the enhancement layer block.

2) *Inter-Layer Motion Prediction:* When BMF is 1 and the co-located reference layer block is inter-coded, then inter-layer motion prediction is used. But when BMF is 0 and the *motion prediction flag* (MPF) is 1 then inter-layer motion prediction is used at the MB partition level. In this mode, the reference layer motion vectors are up-sampled and used to perform motion compensation to obtain the enhancement layer block.

3) *Inter-Layer Residual Prediction:* In this mode, the reference layer residual information is used for the prediction of the residual data of inter-coded blocks in the enhancement layer regardless of the value of BMF or MPF. This is indicated by a *residual-prediction flag* (RPF) having a value of 1.

The aforementioned inter-layer prediction mechanisms are similar for spatial scalability and CGS, except that, for CGS, the up-sampling operations and the inter-layer deblocking of intra-coded reference layer MBs are not performed. In addition, in CGS, the inter-layer intra and residual predictions are performed directly in the transform domain.

3. PROPOSED ARCHITECTURE

The overall proposed system architecture for the H.264 SVC decoder supporting spatial and coarse-grained scalability is shown in Fig. 2.

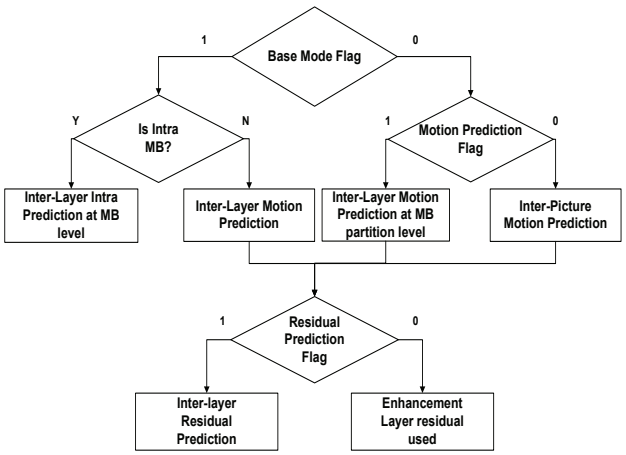

Fig. 1. Decoder enchancement layer mode decision.

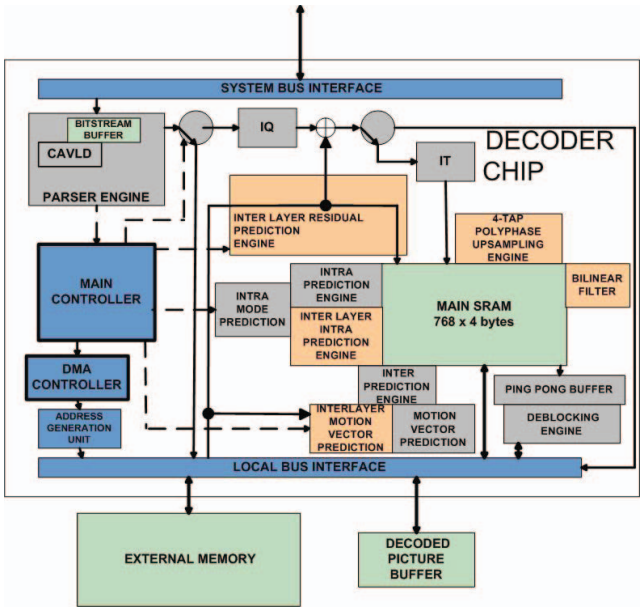

Fig. 2. Proposed H.264/SVC decoder architecture.

The architecture consists of various blocks such as inverse quantization (IQ), inverse transformation (IT), bilinear filtering, intra mode prediction, intra prediction engine, inter prediction engine, motion vector prediction and deblocking engine that are a part of the single-layer H.264 decoder architecture. In addition, the proposed architecture contains blocks that are needed to support spatial scalability and CGS quality scalability in SVC. These SVC-specific blocks include the inter-layer intra prediction engine, the inter-layer residual prediction engine, the inter-layer motion vector prediction, and the four-tap polyphase up-sampling engine.

In Fig. 2, the context adaptive variable length decoder (CAVLD) block which is a part of the parser engine is used to decode the incoming bitstream. The main controller unit takes care of the control flow in the architecture while the direct memory access (DMA) unit along with the address generation unit is used to transfer data between the external memory and the internal memory. A scratch-pad memory (SRAM) is chosen as the internal memory, instead of a cache memory, to reduce power consumption. Since the basic block size of the H.264/AVC video coding standard is 4x4 and only one row of this 4x4 block is frequently accessed at a time, the architecture utilizes a 4-pixel wide (32 bits) data bus and the memory width is also set to be 32 bits. A ping-pong buffer is used to transfer the reconstructed data from the SRAM to the deblocking engine. The decoded picture buffer (DPB) stores the reference pictures used in motion compensation.

4. DATAFLOW & MEMORY REQUIREMENT ANALYSIS

This section describes the dataflow in the proposed architecture and the resulting memory requirement for spatial scalability. Similar analysis holds for CGS except for a few differences which are pointed out later.

The following discussion assumes a bit stream consisting of a base layer (B0) and two enhancement layers (E1 and E2) but this can be easily extended for multiple enhancement layers. To consider the worst-case scenario the slice size is considered equal to the frame size and the target layer to be decoded as E2. The image size is assumed to be 4CIF (704 x 576) at E2, CIF (352 x 288) at E1 and QCIF (176 x 144) at B0.

Dataflow depends on the MB modes and whether the MB belongs to the target layer or the reference layer. In the proposed architecture, for inter-coded MBs, only the target E2 layer blocks are decoded. Also, the residual data for the lower reference layers is inverse quantized, inverse transformed and, if RPF = 1, predicted from their lower layers and finally stored in the external memory to be used for inter-layer residual prediction of the upper layers. On the other hand, all intra-coded blocks are fully decoded due to the high dependency among neighboring and/or co-located intra-coded blocks, whether the blocks are spatially predicted or inter-layer predicted. This is done for intra-coded blocks in order to avoid performing frequent dependency checks, which could substantially increase the number of external memory accesses.

Five cases can be considered:

Case 1 - Intra Coded Blocks: These blocks correspond to intra coded blocks having no dependency on the reference layers. The various intra prediction modes are Intra_16x16, Intra_8x8, Intra_4x4 and I_PCM. Fig 3 shows the flow of data for the luma pixels of the Intra_16x16 mode. As shown in the figure, the MB residual data (R_D) after being inverse quantized and inverse transformed is added to the predicted pixels obtained using the intra prediction engine, to reconstruct the MB (C_D) in the SRAM after which it is deblocked and stored into the external memory to be accessed when required. Additionally if the MB belongs to the target layer and is used as a reference picture then it is also stored in the Decoded Picture Buffer (DPB).

During intra prediction, additional data from the neighboring blocks is required. In the proposed architecture the data from the neighboring blocks is retained in the SRAM as long as necessary. Lifetime analysis is employed to efficiently retain and discard data in order to optimize on-chip memory size. Fig 4 illustrates the dependencies in the case of the Intra_16x16 mode. Each cell represents a 4x4 block and the blocks A-D and W-Z represent MBs. The MBs are decoded in a row-wise raster scan order, i.e., A,B,C,D, followed by W,X,Y,Z. Consider that the MB shaded in grey (block 'X'), whose blocks are marked with '?', is currently being decoded. Blocks A-

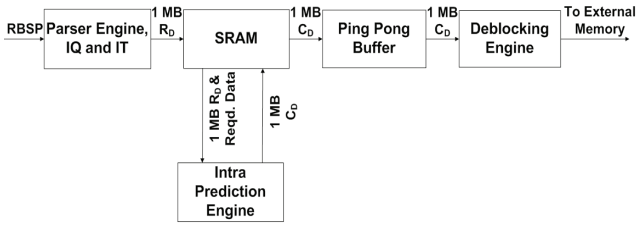

Fig. 3. Dataflow for Intra_16x16 decoding.

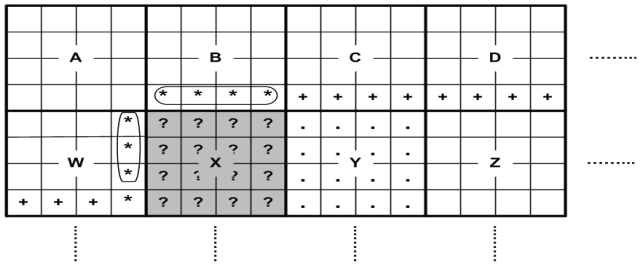

Fig. 4. Intra_16x16 block decoding: spatial dependencies.

W (all blocks above and to the left of X) would have been already decoded at this point. The possible data required for decoding the block X is marked with a '*'. MB X is dependent on the last row of pixels from B and the rightmost column of W. The last row of the blocks marked with '+' is to be retained since they are needed to reconstruct other MBs. The blocks marked with '.', which represent residual data are loaded while block X is being decoded in order for Y to be decoded next. Also the data in the empty blocks of A-W have been discarded because they are no longer required. The blocks that are after Y (e.g., Z in Fig. 4) are empty representing the data that is not yet loaded in the memory. After MB X has been decoded the circled data in Fig 4 can be discarded since it is not required any more.

With the help of the above scheme, for a 4CIF frame, it was found that the amount of memory required for decoding the luma samples of an MB using the Intra_16x16 mode is 1532 bytes. For 4:2:0 sampling, the chroma blocks require half the memory required for the luma blocks. So, the total required memory for decoding a MB is (1532 x 1.5) = 2298 bytes.

The flow of data in the case of the Intra_8x8 and Intra_4x4 modes remains the same as the Intra_16x16 except that, instead of a 16x16 MB, 8x8 and 4x4 blocks of data are considered, respectively. The lifetime analysis is also very similar and varies slightly because of the different prediction modes from the Intra_16x16 mode. The memory requirement for the Intra_8x8 and Intra_4x4 modes was found to be 1440 bytes and 1224 bytes, respectively.

Since the I_PCM mode does not have any dependencies across MBs, its memory requirement is within the limit of the above modes and, hence, no separate analysis is necessary.

Case 2 - Inter-layer Intra Coded Blocks: These blocks correspond to the intra coded blocks which are predicted from the lower reference layer blocks. The dataflow in this case is very similar to the case of intra coded blocks except that the predicted pixels are obtained by up-sampling the reconstructed co-located blocks in the reference layer which are stored in the external memory. In this case, 8x8 blocks are processed at a time so that only 4x4 blocks of the co-located reference layer have to be up-sampled, reducing the load on

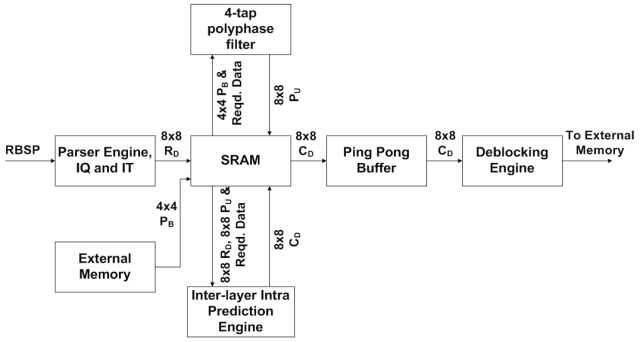

Fig. 5. Dataflow for inter-layer intra decoding.

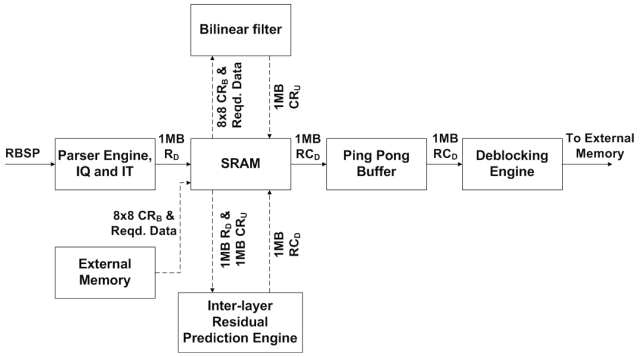

Fig. 6. Dataflow for decoding of reference layer inter-coded data.

the SRAM.

Fig. 5 shows the flow of data for the luma pixels of the interlayer intra coded blocks. The 4x4 reference layer block (P_B) is upsampled using a 4-tap polyphase filter to obtain an 8x8 prediction (P_U). The inter-layer intra prediction engine is then used to reconstruct the 8x8 block (C_D) which is further deblocked and stored in the external memory. In the case of chroma pixels, a bilinear filter is used instead of the 4-tap polyphase filter to upsample the co-located reference layer blocks.

Using a similar lifetime analysis as in Case 1, the memory requirement for inter-layer intra coding was found to be 2800 bytes for 4CIF resolution.

Case 3 - Inter Coded Blocks in the Reference Layers: These blocks correspond to the inter-coded blocks, including inter-picture coded or inter-layer coded blocks, and belong to the reference layers.

Fig. 6 shows the dataflow in this case. The MB residual data is inverse quantized, inverse transformed and, if RPF = 1, the result (R_D) is added using the inter layer residual prediction engine to the residual data (CR_U). CR_U is obtained by up-sampling the colocated reference layer data (CR_B) which is accessed from external memory. The result (RC_D) is stored into the external memory. Conditional up-sampling of the residual data is shown by the dotted lines in Fig. 6. Regardless of the value of the RPF flag, the motion vectors are predicted using the motion vector prediction block (not shown in Fig. 6) and then stored into the external memory.

Case 4 - Inter-Picture Motion Predicted Blocks in Target Layer: These blocks correspond to the inter-picture coded blocks which belong to the target layer and have to be decoded completely. The motion vectors are not predicted from the reference layers. As shown

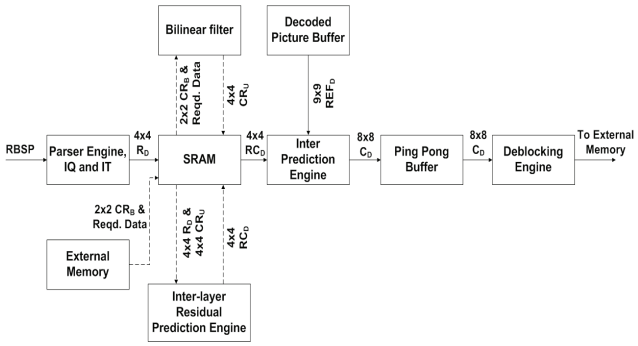

Fig. 7. Dataflow for decoding target layer inter-picture motion prediction blocks.

in Fig. 7, the dataflow remains the same as discussed in Case 3 until the stage at which the reconstructed residual data (RC_D) is obtained. The additional stage in this case is the motion compensation (MC) stage. The motion vectors are predicted using the motion vector prediction block (not shown in Fig.7) and then used for MC using the Inter Prediction Engine. The reference picture data (REF_D) for MC is obtained from the DPB. The reconstructed data (C_D) is then deblocked and stored into the external memory.

Case 5 - Inter-layer Motion Predicted Blocks in the Target Layer: These blocks correspond to inter-layer coded blocks for which the motion vectors are predicted from the reference layers and the blocks belong to the target layer. The dataflow for these blocks is similar to Case 4 except for the fact that the motion vectors are obtained from the up-sampled motion vectors of the co-located reference layer blocks, which are stored in the external memory, using the inter-layer motion vector prediction block.

For the inter-prediction modes (*Cases 4* and *5*), the worst case occurs during bi-directional prediction for a 16x16 macroblock. In the proposed architecture, MC for a 16x16 block is performed by splitting it into 8x8 blocks, to reduce the memory requirement. The maximum memory size required for MC (bi-directional) of 8x8 blocks using quarter pixel interpolation is 2240 bytes.

The dataflow remains almost the same in the case of CGS except for the fact that there is no up-sampling or deblocking of reference layer data. Also, the prediction is done in the transform domain and, thus, the residual data belonging to the reference layers is not inverse transformed until it is needed for decoding the target layer MBs using inter-layer residual prediction. In this latter case, the inverse quantized residual data of the reference layer is added to the inverse quantized residual data of the target layer and then inverse transformed. Table 1 summarizes the memory requirement for various target layer resolutions.

From the above discussions it should be noted that while processing a particular layer of the H.264 SVC bitstream, segments of the decoded data that are required for decoding further portions of the layer are retained in the memory. At the same time the data that is not required for further decoding is discarded to ensure that the size of the on-chip memory remains small. Also, all the intra coded blocks are decoded even if they do not belong to the target layer. This reduces the number of external memory accesses in the case when inter-layer intra prediction is necessary and the co-located reference layer blocks are spatially (intra) predicted from their neighboring blocks. This reduction in external memory accesses translates to a reduction in the power consumption. Hence the proposed architecture is successful in achieving the dual goals of reducing both the on-chip memory and the power consuming external memory accesses.

Table 1. Memory requirement for various frame resolutions

Target Frame Resolution	SRAM Memory Size (worst case in bytes)
QCIF (176 x 144)	1232
CIF (352 x 288)	1568
4CIF (704 x 576)	2800

5. CONCLUSION

In this paper, a system level architecture is proposed for decoding H.264 SVC spatially scalable and CGS bitstreams. The proposed architecture aims at reducing the on-chip memory size while minimizing the number of memory accesses. The proposed architecture requires a 3 KB scratchpad memory for combined (spatial and CGS) scalability bit streams for a target resolution of 4CIF. A 32-bit bus interface has been proposed to access 4 pixels (rows of a 4x4 block) at a time. Further work needs to be done to extend this architecture to support MGS scalability.

6. REFERENCES

[1] T. Wiegand, G. J. Sullivan, J. Reichel, H. Schwarz, and M. Wien, "Joint Draft 11 of SVC Amendment," *Joint Video Team, Doc. JVT-X201*, July 2007.

[2] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sept. 2007.

[3] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC," *in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050*, 2003.

[4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, July 2003.

[5] J. Ribas-Corbera, P. A. Chou, and S. L. A. Regunathan, "Generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 674–687, July 2003.

[6] H.-Y. Kang, K.-A. Jeong, J.-Y. Bae, Y.-S. Lee, and S.-H. Lee, "MPEG4 AVC/H.264 decoder with scalable bus architecture and dual memory controller," *Proc. IEEE ISCAS'04*, vol. 2, pp. 145–148, May 2004.

[7] T.-C. Chen, C.-J. Lian, and L.-G. Chen, "Hardware architecture design of an H.264/AVC video codec," in *Proc. IEEE ASP-DAC*, Jan 2006.

[8] Z. Wei, K. L. Tang, and K. N. Ngan, "Implementation of H.264 on mobile device," *IEEE Trans. Cons. Elect.*, vol. 53, no. 3, pp. 1109–1116, 2007.

[9] P. Dang, "High performance architecture of an application specific processor for the H.264 deblocking filter," *IEEE Transactions on VLSI Systems*, vol. 16, no. 10, pp. 1321–1334, Oct 2008.