A BIT-SERIAL ARCHITECTURE FOR H.264/AVC INTERFRAME DECODING

Paweł Garstecki, Adam Łuczak and Tomasz Żernicki

Division of Multimedia Telecommunications and Radioelectronics,
Poznan University of Technology
Piotrowo 3a, 60-695 Poznan, Poland
[pgarstec, aluczak, tzernicki] @multimedia.edu.pl
web: www.multimedia.edu.pl

ABSTRACT

Abstract— The H.264/AVC is the most recent standard of video compression. In this paper, an original and efficient architecture of inter prediction block in an H.264/AVC decoder is presented. It is shown that the bit-serial arithmetic can be successfully used for interpolation filter implementation and the resulting architecture is fully pipelined. The inter prediction module was implemented in Verilog HDL and synthesized and then tested on Xilinx Virtex IV family devices. The simulation results indicate that the proposed bit-serial architecture of interpolation filter is very efficient and clock frequency close to the image sampling frequency is enough to perform image reconstruction.

1. INTRODUCTION

The H264/AVC [1,2] is a new video coding standard. Its compression efficiency is much better than any previous technique like MPEG4 or H.263. Due to high complexity of tools used by AVC/H.264 standard [3,4], high clock frequencies of hardware decoders (both pure hardware and DSP based structures) are required. However, it causes higher power consumption, which can be an obstacle for some applications e.g. mobile devices. Therefore, in order to achieve low power consumption it is necessary to develop new structures of decoder's blocks.

The AVC decoder consists of stream parser block, management unit and reconstruction module. The most complex part of the decoder is image reconstruction. During the image reconstruction process an intra-frame (intra) or an inter-frame (inter) prediction as well as an inverse integer transform of prediction error are carried out.

In this paper a bit-serial architecture of the inter prediction module is presented. The proposed structure of reconstruction entity (shown in Fig.1) consists of blocks of inter and intra predictors, an inverse transformation module, local buffers (cache and context) and two fifo queues. Fifo queues contain coefficients (coeff fifo) and control data e.g. prediction modes and motion vectors (command fifo). This modules are universal interfaces of reconstruction module and contain data loaded from a hardware parser or a processor unit. The proposed reconstruction block enables reconstruction of a frame of a sequence encoded in any possible mode. A bit-serial architecture for the intra image prediction and transformation module was shown in [5]. It is proved that the bit-serial structure may as efficient as the parallel one and it seems to be more suitable for implementing in FPGA structures. It is because of utilizing of local connections mostly, narrow data busses and simple logic functions (e.g. serial adders).

An interpolation process for luminance and chrominance samples uses different algorithms. The most complex part of inter prediction in AVC is luminance samples prediction due to numerous prediction modes and advanced interpolation algorithm. Therefore, it is the main part of this paper.

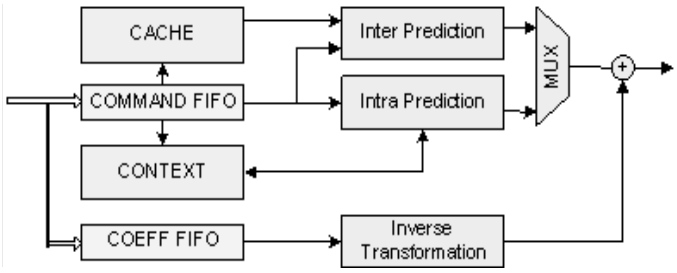

Figure 1. AVC reconstruction block diagram.

2. THE H.264/AVC INTER PREDICTION

The H.264/AVC inter prediction can be invoked in four main modes: Inter16x16, Inter16x8, Inter8x16 and Inter8x8. Each mode defines different partitioning of a macroblock as shown in Fig.2. Furthermore, Inter8x8 prediction mode block can be split into four 4x4 blocks and a separate motion vector can be defined for each one of such elements. The AVC defines accuracy of motion vector estimation as well. The maximal accuracy of the inter prediction algorithm for luminance samples is a quarter of a sample distance (a quarter-pel interpolation). The AVC standard allows for using one-directional or two-directional inter prediction of each image block.

In the decoder it is necessary to perform an inverse process to reconstruct the original image samples. This is achieved by invoking an interpolation algorithm.

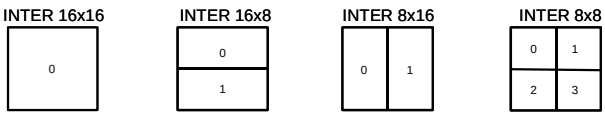

Figure 2. A macroblock partitioning for AVC main prediction modes.

The samples' interpolation is done in a different way depending on motion vectors values. Interpolation is a very complex filtering problem and requires a lot of data to be fetched from memory. In general, in order to start filtering process of a single 4x4 block it is necessary to load values of one 4x4 block and its surrounding (Fig.3) from a reference image. This part of the algorithm depends on macroblock type and motion vectors' values that point a reference image area in a reference pictures. The number of samples that must be loaded to interpolate single 4x4 image block (16 samples) is up to 81 per each motion vector. Accordingly to the AVC standard it is possible to assign up to two motion vectors to each 4x4 luminance block. Since there are 16 blocks in a macroblock and 1602 macroblocks in 4CIF (720x576) or SDTV (704x576) image the number of samples that must be loaded from external memory is 2*81*16*1620*16 = 4 199 040 samples per each frame.

This results in considerable transfers of data from external memory up to 100 MB/s in the case of 25 Hz 4CIF sequence. It was shown that this can be a serious problem and a special structures for data acquisition may be required[6].

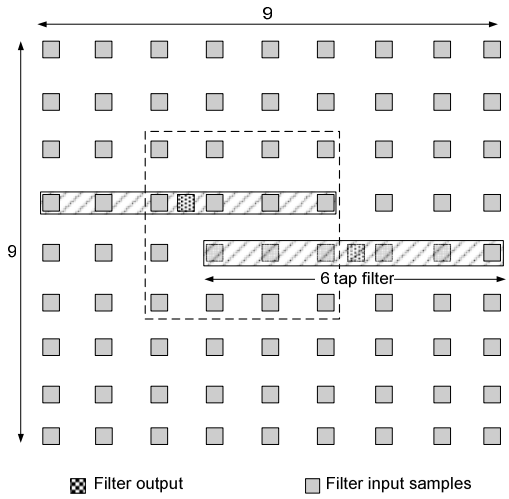

Figure 3. Luminance samples interpolation using 6-tap filter.

The other stage of the algorithm is filtering of samples of 4x4 block and output values computation. The prediction values for a half and a quarter-pel positions are calculated by applying 6-tap filter defined by equation (1) and simple bilinear filter.

$$x(n) = x(n-5) - 5\cdot x(n-3) + 20\cdot x(n-1) + \\ + 20\cdot x(n+1) - 5\cdot x(n+3) + x(n+5) \quad (1)$$

The filtering process can be carried out in vertical or/and horizontal directions. In the case of half-pel interpolation this process is invoked once but when the output sample is a quarter-position sample it is necessary to perform additional filtering.

A separate issue is chrominance samples interpolation. An output value is a weighted sum of nearest four full-sample location values and it is defined be equation (2).

$$y(x_C, y_C) = (8-F_{CX})(8-F_{CY})\cdot A + F_{CX}(8-F_{CY})\cdot B + \\ + (8-F_{CX})F_{CY}\cdot C + F_{CX}F_{CY}\cdot D \quad (2)$$

In Fig.4 an example of chrominance samples interpolation is shown. The weight coefficients F_{CX} and F_{CY} are defined by the standard and depend on output sample's position.

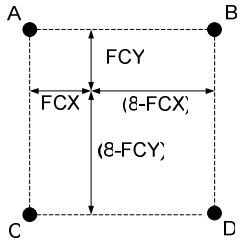

Figure 4. Chrominance samples interpolation.

3. THE ARCHITECTURE

The proposed architecture of an inter prediction block is shown in Fig.4 The two main parts of this entity are a local cache memory with data pre-fetch and a filters' matrix.

The filtering process is performed in two stages for each 4x4 image block. The steps of this algorithm are: data loading from cache memory into a buffer and filtering. The filtering algorithm is always carried in 4x4 block regardless the mode used.

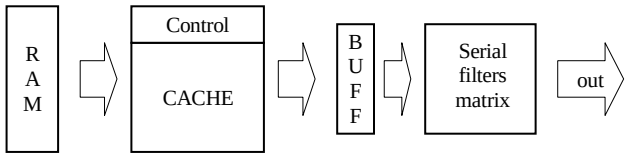

Figure 5. Inter-frame prediction with serial filtering.

3.1 Data pre-fetch

Before interpolation process can be invoked all the necessary data must be transferred from extern memory into local cache. The main purpose of using dedicated pre-fetch module is minimizing number of an access cycles and delay caused by address switching in DDR or SDR RAM memories. In order to achieve this attention was paid to data alignment and method of data fetching.

In the proposed structure the cache memory is a matrix of 6x6 blocks (24x24 samples). In each step of the algorithm the cache memory is updated accordingly to current macroblock type and prediction mode. The module acquires samples that are aligned in 32-bit words and ordered linearly in memory within for a 4x4 image block. Additionally, a number of reading modes is used to minimize the amount of data transferred from RAM memory. As already mentioned, there can be up to sixteen motion vectors per macroblock. In the worst case, when each 4x4 image block was encoded separately a lot of data must be read to perform the interpolation process. However, when bigger partitions are used it is possible to use previously read data and introduce pipelining.

The macroblock partitioning modes are not equally probable[7]. It was shown that this can be exploiting in construction of a fast inter mode selection algorithms as well [8]. Therefore, the proposed algorithm uses four modes illustrated in Fig.6. Mode *M0* is the basic mode that is used in most cases. It allows for reading 4x4 reference block and its surrounding. The remaining modes (*M1*, *M2* and *M3*) are exploiting when transferring additional data for bigger partitions. Some examples of data reading are presented in Fig.7. Cache filling and filtering operations are pipelined. When all data for a single 4x4 block are already loaded into cache memory the interpolation process is started. At the same time data acquirement for the next block is invoked. In Fig.8 a time diagram of example process is shown (case of Intra_8x8). Data loading is carried out in three stages in this mode. A single 4x4 block and all surrounding blocks are loaded first (first stage: 9 blocks). Next, the first block interpolation is started and second stage data are loaded into cache (second stage: 3 + 3 blocks).

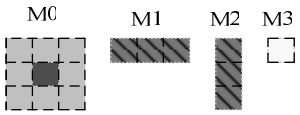

Figure 6. Memory reading modes (cache update).

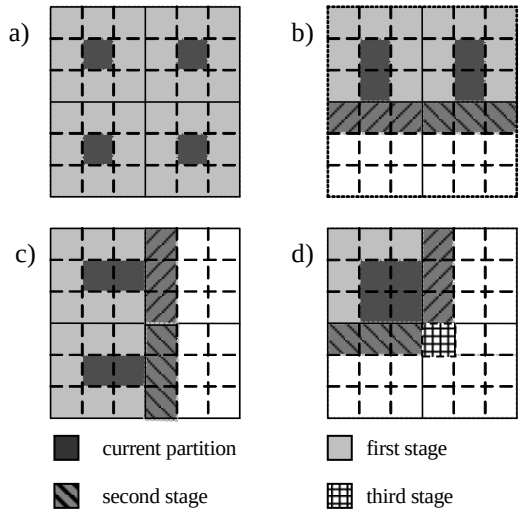

Figure 7. Examples of applying reading modes and cache memory partitioning (a – Intra4x4, b – Intra4x8, c –Intra8x4, d – Intra8x8).

After completing all operations the third stage of data acquisition is carried out (1 block). Finally, when all data required for the first 8x8 block are stored in cache loading for another image partition can be started (second part filling in Fig.8).

The same module is also used in the interpolation process of chrominance samples. In this case fetching is much less complex because only four reference samples are required to calculate an output sample. To interpolate a 4x4

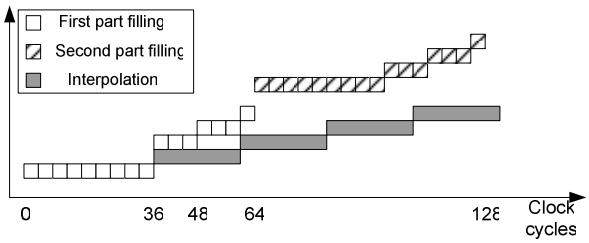

Figure 8. Time diagram of local cache filling (Intra8x8 mode).

chrominance block 9 reference samples must be read. Therefore, if only a single chrominance block is read the interpolation process may be started.

3.2 Luminance filtering block

The bit-serial AVC filter structure is shown in Fig.9. It is a very simple entity that is composed of four serial adder sections and four delays and is implemented in bit-serial arithmetic.

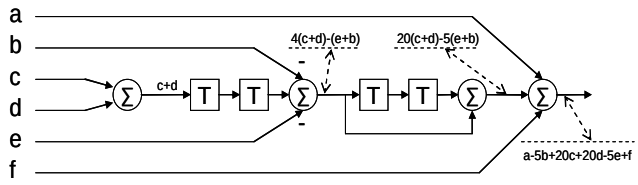

Figure 9. Inter-frame prediction with serial filtering Σ- serial adder, T- delay)

Taking into account simplicity of this filter it is possible to implement a matrix of filters that allow for interpolating all samples within a 4x4 image block simultaneously.

The structure of serial filtering matrix is a 9x9 matrix containing 36 vertical and 36 horizontal filters. The allocation of filters is shown in Fig.10. This filters set is used to compute all partial values and sixteen output samples' values within an image block. An advantage of this structure is relatively small portion of data that must be read from the memory. To interpolate one sample 36 data samples must be read from the cache memory; however, to interpolate all sixteen samples in a block, the structure requires only 81 reference values to be loaded. That is because the data samples can be shared between several filters.

Some filters in the proposed matrix are reduced because several coefficients are common for vertical and horizontal processing (grey squares in Fig.10). This fact causes the reduction of the structure. The decrease of area in comparison to size of full structure containing 72 filters is about 50%.

Depending on the filtering mode the result is selected from among 72 filters. For central half-pel samples interpolation the result is combined with the use of additional sixteen filters that process previously filtered data.

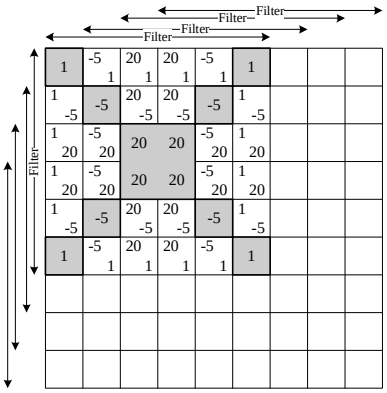

Figure 10. Serial filters matrix.

3.3 Chrominance filtering block

The architecture of a chrominance filtering block is presented in Fig.11. The structure contains four simple multipliers that are implemented as parallel 6-bit accumulators (ACC) and output accumulator (DA-ACC). The output accumulator is implemented in distributed arithmetic.

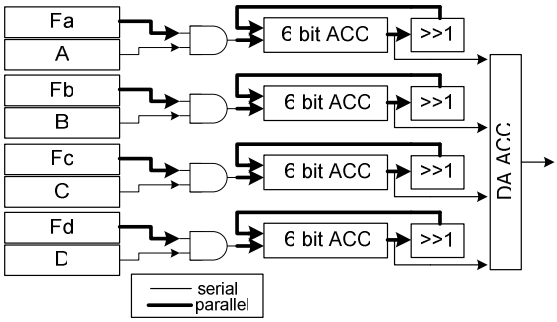

Figure 11. Structure of chrominance interpolation filter (*A,…, D* – reference samples, *Fa, …, Fd* – weight coefficients, ACC – accumulator, DA-ACC – output accumulator).

Four chrominance samples are calculated simultaneously; therefore the matrix contains four simple filters. The reference values *A, …, D* are the nearest full position chrominance samples.

Weight coefficients *Fa, …, Fd* are calculated as products of values defined by the standard and they are constant within a single chrominance block. Therefore, these values are computed during samples loading stage (caching) in advance prior to filtering stage and a single multiplying unit is used. In this way the structure is reduced and the filter's structure shown in Fig.11 uses previously prepared weights.

4. SYNTHESIS RESULTS

The proposed architecture has been implemented in order to fulfil requirements of low transfers from memory and efficient data processing.

The structure was also synthesized with Xilinx ISE tool for Virtex2 and Virtex4 family devices. The synthesis results are presented in table 1. The area of cache module is 374 FPGA slices and the maximal operating frequency is about 198MHz for *Xilinx Virtex IV* (150MHz – *Viretx II*). The area occupancy of filters matrix is 610 slices and the maximal operating frequency is 534MHz for *Virtex IV* (about 350 MHz for *Virtex II*).

The chrominance interpolation module maximal operating frequency is 203 MHz and the area usage is 94 slices.

Table 1. Synthesis results for Xilinx Virtex IV.

Module	*Cache*	*Luminance filters matrix*	*Chrominance filters*
Area occupancy [FPGA slices]	374	610	94
Maximal operating frequency [MHz]	198,1	534,9	203,5

The implemented module is relatively small structure and can operate at a high clock frequency thus high processing efficiency is achieved.

The small data transfers are achieved for larger partitions (the lowest in the case of *Inter_16x16* mode). Taking into account statistics of intra prediction modes (the most probable partition sizes are 16x16 and 8x8) it can be stated that the average required data transfer for one-directional interpolation is about 35MB/s (it is less than half of the worst case transfer). The average transfer for the two directional-interpolation is about 55MB/s (because of different statistics of interpolation modes usage).

Table 2. Data transfers for one-directional inter prediction (single port memory, 32-bit words)

Macroblock type		*Number of reference samples (32-bit words)*	*Required data transfer (4CIF, 25Hz)*[1] *[MB/s]*
Mode (partition size)	*Number of partitions per macroblock*		
16x16	1	576 (144)	22,25
16x8 (8x16)	2	768 (192)	29,66
8x8	4	1024 (256)	39,55
4x8 (8x4)	8	1536 (384)	59,33
4x4	16	2304 (576)	88,99

In Fig.12 a time diagram of interpolation procedure is shown. The loading stage consists in reading 81 data samples thus it requires 81 clock cycles. The filtering stage and output values computation requires about 44 clock cycles. Summing up, the processing time of a single 4x4 image block is 125 clock cycles. However, due to pipelining that is exploited the macroblock processing time is 1357 clock cycles. This is about 5,3 clock cycle per sample. An improved version of the presented module uses double port memories and this allows for reducing required clock frequency (data loading time is 50% shorter). The processing efficiency in this case is 754 clock cycles per macroblock and this is about 3 cycles per sample.

[1] *In the case of two-directional image prediction the required transfer is doubled*

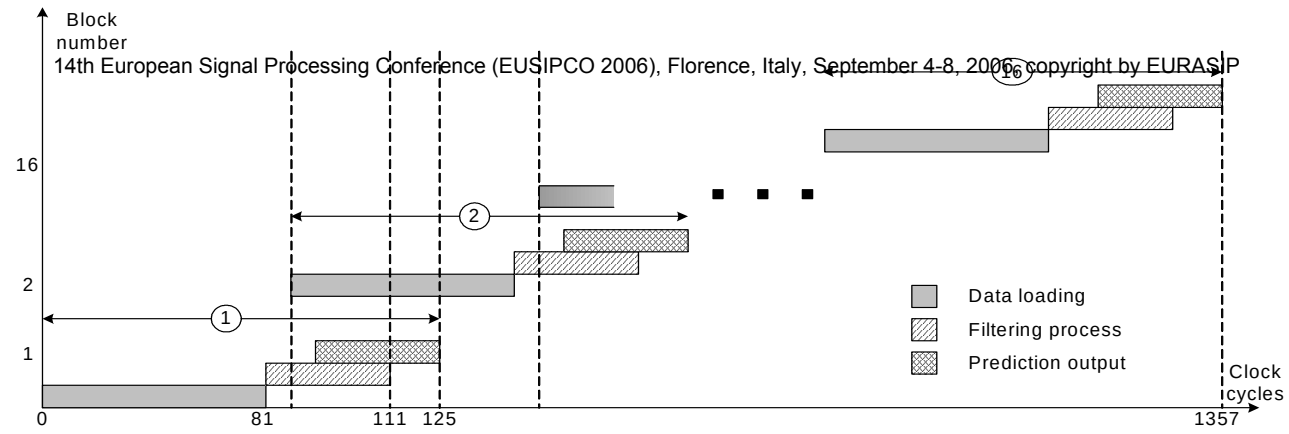

Figure 12. Macroblock filtering efficiency (single port cache memory, 16 samples

Interpolation of chrominance samples is carried out within 16 clock cycles.

5. CONCLUSIONS

Original architecture of inter prediction block for H.264/AVC decoder has been presented. Software simulations and tests on FPGA device (Virtex) confirm that described bit-serial architecture is suitable for FPGA devices. Moreover, proposed design achieves high performance with relatively low clock frequencies.

Described module of inter prediction is an independent module and can be used as a part of hardware decoder or as a hardware accelerator for processor based decoder as well.

ACKNOWLEDGMENT

The work was supported by the public founds as a research project.

REFERENCES

[1] ISO/IEC~JTC~1/SC~29/WG~11, “ISO/IEC 14496 10 Advanced Video Coding “, Redmond, July 2003

[2] T. Wiegand, G.J. Sullivan, G. Björntegaard, A. Luthra, “Overview of the H.264/AVC video coding standard”, IEEE. Trans. on Circuits and Systems for Video Technology, vol. 13, pp. 560—576, July 2003

[3] M. Horovitz, A. Joch, F. Kossentini and A. Hallapuro, “H.264/AVC Baseline Profile Decoder Complexity Analysis”, IEEE. Trans. on Circuits and Systems for Video

Technology, vol. 13, pp. 704—716, July 2003

[4] V. Lappalainen, A. Hallapuro and T.D. Hämäläinen, “Complexity of Optimized H.26L Video Decoder Implementation”, IEEE. Trans. on Circuits and Systems for Video Technology, vol. 13, pp. 717—725, July 2003

[5] P. Garstecki, A. Łuczak, “A flexible architecture for image reconstruction in H.264/AVC decoders”, In proc. of 17th ECCTD’05, Cork, Ireland, vol. I, pp. 217-220, August 2005

[6] H.-Y Kang, K.-A. Jeong, J.-Y. Bae, Y.-S. Lee, S.-H.Lee, “MPEG4 AVC/H.264 decoder with scalable bus architecture and dual memory controller”, In Proc of International Symposium on Circuits and Systems, Vancouver, Canada, vol.2, pp. 145-148, May 2004.

[7] T. Dzięcielewski, T. Grajek, J. Marek, „Experimental analysis of encoding modes statisctics in advanced video coding”, X Poznan Telecommunication Workshop, Poznań, Poland, pp. 115-120, 2005 (*Polish version only*)

[8] Z. Zhou, M.-T. Sun, “Fast macroblock inter mode decision and motion estimation for H.264/MPEG-4 AVC”, In proc. of ICIP’04, Singapore, vol. 2, pp. 789-792, October 2004.