AREA-TIME TRADEOFFS IN H.264/AVC DEBLOCKING FILTER DESIGN FOR MOBILE DEVICES

Vladimir Ciric, Ivan Milentijevic
Faculty of Electronic Engineering, University of Nis, Serbia
{vciric, milentijevic}@elfak.ni.ac.yu

ABSTRACT

H.264/AVC is a new international standard for the compression of natural video images, in which deblocking filter has been adopted to remove blocking artifacts. The deblocking filter is either time or area consuming system component. The goal of this paper is a design of area-time efficient H.264/AVC deblocking filter suitable for application in mobile devices. Area-time tradeoff is enabled by using configurable folded bit-plane filter as a core for deblocking filter implementation. Folded filter implementation parameters are obtained in the manner of achievement of minimal area consumption, keeping in mind throughput requirement in mobile devices. System's architecture is presented in detail, as well as the architecture of folded deblocking filter. With aim to illustrate functionality and tradeoffs related to occupation of chip resources and achieved throughputs we present results of FPGA prototyping. Proposed deblocking filter requires extremely low gate count still meeting the mobile application throughput requirements.

1. INTRODUCTION

Mobile devices technology is changing rapidly. There is an increasing number of wireless-communications standards, code-division multiple access, and emerging third-generation technologies. However, as wireless technologies mature, service providers differentiate themselves by offering new features, such as multimedia capabilities [1]. Video coding applications such as H.264/MPEG-4 Advanced Video Coding, adopts filter called deblocking filter in order to eliminate blocking artifacts and to achieve a better coding efficiency [2],[3]. The deblocking filter is much more complex than common low-pass FIR filters, thus in mobile computing, having in mind computational power, deblocking on general purpose processor is a waste of resources. Deblocking filter is usually implemented as a filter bank, where different filter is selected in the dependence on blurring strength on image edge. Implementation of a filter bank adds cost, takes up space and increases power usage in mobile devices. This problem can be solved using configurable architecture, letting a single architecture to perform different computations [1],[4],[5].

In order to attain high performance in filter design, parallel implementation strategies such as systolic methods have been applied. Thus, due to their geometrical regularity, they are suitable for VLSI implementations, either as stand-alone modules or as a part of complex digital data path. The choice of structure for the implementation of an FIR filter includes consideration of the factors such as hardware complexity and throughput. Many different structures exist, most of which provide some tradeoff between complexity and throughput [6]. For dedicated applications, the design choice then becomes the minimal complexity structure that can achieve a required throughput rate. Therefore, in order to establish optimal area-time tradeoff, a careful choice of circuit design style is necessary [7].

The goal of this paper is a design of area-time efficient H.264/AVC deblocking filter suitable for application in mobile devices. Area-time tradeoff will be exploited by using a configurable folded semi-systolic bit-plane filter array [5] as a core for deblocking filter implementation. Filter implementation parameters will be obtained with aim to achieve minimal area consumption, keeping in mind throughput requirement in mobile devices. System's architecture will be presented in detail, as well as the architecture of folded deblocking filter. With aim to illustrate functionality and efficiency related to tradeoffs on chip resources occupation and achieved throughputs we present results of FPGA prototyping.

The paper is organized as follows: Section 2 gives a background of H.264/AVC deblocking; Section 3 is devoted to principles of configurable FIR filtering on folded bit-plane array; Section 4 is the main section and presents deblocking filter tradeoffs according to configuration abilities of the implemented filter; Section 5 is devoted to implementation results and comparisons, while in Section 6 concluding remarks are given.

2. H.264/AVC DEBLOCKING ALGORITHM

With aim to clarify the design tradeoffs of folded H.264/AVC deblocking filter we give a brief review of deblocking algorithm.

The functional blocks of H.264/AVC encoder and decoder are shown in Fig. 1 and Fig. 2, respectively. The transformation algorithm adopted by H.264/AVC (block T in Figs. 1 and 2) is 4x4 Discrete Cosine Transform (DCT). In the DCT-based standards annoying blocking artifacts arise when compression ratio is high (i.e. video

1-4244-0779-6/07/$20.00 ©2007 IEEE

conferencing using cellular phones). The effect is caused by non-overlapping nature of the blocked-base DCT coding and quantization of coefficients [2]. H.264/AVC adopts deblocking filter into the coding loop (Fig. 1 and Fig. 2) to remove blocking artifacts and to achieve much better subjective visual effect, as well as coding efficiency [2],[3].

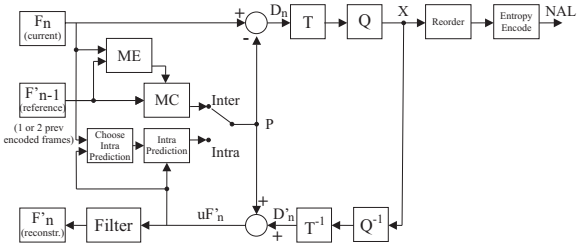

Figure 1. H.264/AVC Encoder

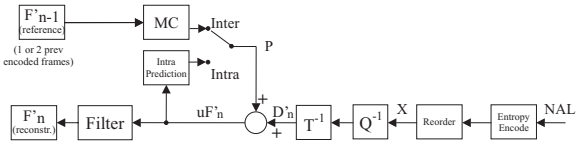

Figure 2. H.264/AVC Decoder

Deblocking filter is applied to all 4x4-block edges, except edges at the boundary of the picture. The filtering is performed on 16x16 pixels macro blocks (MB), one after another, with all MBs of a picture in raster scan order [2]. The deblocking filter is invoked for luma and chroma component separately. For each MB, vertical edges are filtered from left to right, and horizontal edges are filtered from top to bottom (Fig. 3a, Fig. 3b), [2],[3],[9].

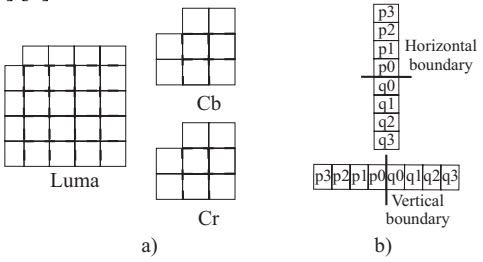

Figure 3. H.264/AVC macro block, a) Edges filter order in 16x16 MB; b) Pixels across boundaries

In deblocking, the output depends on the boundary strength and the gradient of image samples across the boundary (Fig. 3b), [2],[3]. As proposed in [3], there are five filtering modes: mode 4, strongest filter (7 tap filter affecting 8 pixels); mode 3, strong filter (7 tap filter affecting 6 pixels); mode 1 and 2 (5 tap filter affecting 5 pixels); mode 0 (5 tap filter affecting 4 pixels, or 3 tap filter affecting 2 pixels).

3. CONFIGURABLE FIR FILTERING ON FOLDED BIT-PLANE ARRAY

Output words $\{y_i\}$ FIR filter are computed as

$$y_i = c_0 x_i + c_1 x_{i-1} + \cdots + c_{k-1} x_{i-k+1} \quad (1)$$

where $c_0, c_1, \ldots, c_{k-1}$ are coefficients while $\{x_i\}$ are input words. Computation (1) can be realized in different manners. When high performances are required systolic arrays are frequently used.

Folding [7] or time-multiplexing is a technique for efficient resource sharing for area-constrained behavioral synthesis from a data-flow graph (DFG). The basic terms of folding technique are folding set, folding factor and folding order. Folding set (S) is defined as an ordered set of operations, which contains N operations, time-multiplexed on the same functional unit (FU). The number of operations executed by the same FU is called folding factor (N). Folding order (v) of operation H_v is a time instance in which FU of folded system executes the operation [7].

The following notation is adopted: k_C – number of taps, m_C – coefficient length, c_i^j – bit of coefficient c_i (with weight 2^j), N – folding factor, k – number of folding sets (FUs of folded system), n – input word length, y – output word length, L – total number of operations for computation of one output word in (1).

The **C**onfigurable **F**olded bit-plane **F**IR **F**ilter (C3F) is semi-systolic architecture shown in Fig. 4 [5],[10], obtained by application of folding technique to semi-systolic bit-plane FIR filter. Applying the folding technique to the bit-plane FIR filter architecture the number of basic cells is reduced for factor N, at the cost of decreasing the throughput for the same factor. Thus, finding suitable area-time tradeoff factor lies in finding suitable folding factor N.

The data flow of the C3F for the case k=3, N=4, k_C=2 and m_C=6 is given in Fig. 5. Each row of basic cells in Fig. 4 stands for one FU (i.e. folding set) that performs

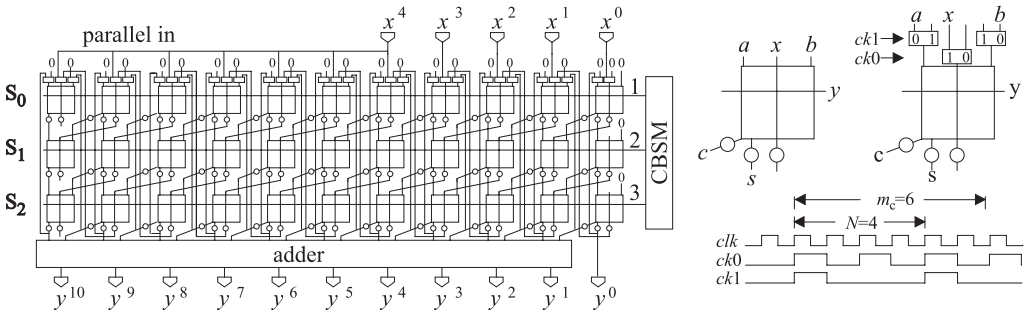

Figure 4. Functional block diagram of C3F with configurable number of coefficients and coefficient length for k=3 and N=4

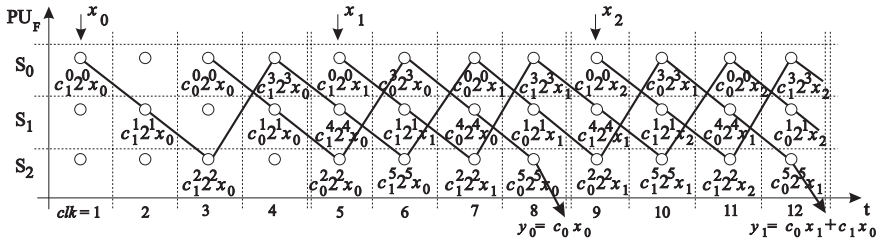

Figure 5. Data flow of C3F (k=3, N=4, k_C =2 and m_C=6)

multiplication of input word with coefficient bit c_i^j, and addition of formed partial product to previously computed sum of partial products. Partial product, computed in the FU S_i (i=0,1,…,k-2), enters the FU S_{i+1} with delay of one clock cycle (Fig. 5), together with the input word shifted for one position to the left (multiplication by 2^1), Fig. 4 and Fig. 5. The computation of new output word starts in each N-th time instance in FU S_0 (Fig. 5). Coefficient Bit Supply Module (CBSM), Fig. 4, implements coefficient reordering algorithm [10] in order to enter coefficient bits in proper order (Fig. 5).

4. CONFIGURABILITY AND DEBLOCKING FILTER PARAMETERS

Each output word (1) is available at the FU S_{k-1} after

$$L=k_C \cdot m_C = k \cdot N \quad (2)$$

time instances (Fig. 5). The application of folding technique, described in [10], brings the constraint that the number of taps can not be equal to the number of FUs, i.e.

$$k_C \neq k. \quad (3)$$

The architecture from Fig. 4 is designed to let the number of coefficients (taps), k_C, to vary at the cost of coefficient length (m_C), keeping the overall number of computations (L) constant, and equal to the product of array dimensions $k \cdot N$, as it is given in eq. (2). The idea on configuration of number of taps lies in using folding technique to fold array and obtain operation ordering rules in data path from Fig. 5 [10]. Thus, configuration can be performed by proper entering of input words and coefficient bits. In array from Fig. 4 the number of taps (k_C) and coefficient length (m_C) are controlled by signals $ck0$ and $ck1$.

Further more, as it is described in [10], folding factor N effects only CBSM reordering algorithm, while doesn't array size, and therefore it can easily be reduced. In that manner same array performs computations faster, according to (2), at the cost of either number of taps or coefficient length. Modified CBSM is called CBSM+.

As proposed in [3], depending on boundary strength, five filtering modes with following coefficients can be used in deblocking: {1,1,1,2,1,1,1}/8 for mode 4 and 3, {1,1,4,1,1}/8 for modes 2 and 1, either {1,1,4,1,1}/8 or {1,2,1}/4 for mode 0. Total number of required operations, L, in respect to (2) for modes 4 and 3 can be obtained by multiplying m_C, $\max(c_i) \leq 2^{m_C} \Rightarrow m_C=2$, and the number of taps k_C=7, resulting in L=14. In the same manner it can be shown that modes 2 and 1 requires L=15, while mode 0 requires L=15 or L=6 operations. In respect to (2), even with ability of changing folding factor N, it is not possible to find k and N pair that can implement 7-tap filter with m_C=2, and 5-tap with m_C=3 filters on the same array. Thus, for mod 4 and 3 filters coefficient length is extended to m_C=3, where L=21. Hence the minimal suitable array size to cover all previously mentioned filtering modes is k=3, while maximal folding factor is N_{MAX}=7. However, due to (3), 3 tap mode 0 filter must be realized as 4 tap filter with c_3=0. Mode 0 filter has extended coefficient length from m_C=2 to m_C=3, in order to fit computation in k=3, and N=4. Configuration abilities for the chosen folded array, along with initial latency and clock cycles needed for reconfiguration are given in Table 1.

5. DEBLOCKING FILTER IMPLEMENTATION AND COMAPRISONS

Deblocking filter architecture is shown in Fig. 6. Filter consists of C3F as a core, which is surrounded with two on-chip RAM modules for pixel blocks P and Q, and control unit. On-chip RAM is temporary storage for image MBs, while control unit provides proper ordering of p and q pixel stream to C3F.

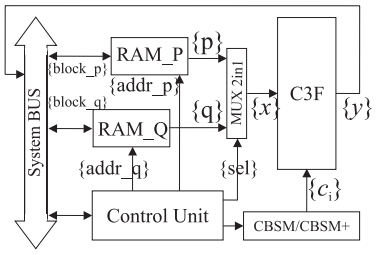

Figure 6. Deblocking filter architecture

To evaluate the accuracy and efficiency of the proposed architecture we described the proposed design in VHDL, at RTL level, which is synthesizable. We have evaluated mentioned folded array for two cases. The first is with CBSM, while the other is with CBSM+. It should be noted that the reducible folding factor is employed in the second case. Implementation results of proposed array together with CBSM+, that relates to chip resources occupation and achieved throughput rates, are given in Table 1. On-chip RAMs are not taken into consideration.

According to the JVT verification model [11], a C-program model of deblocking filter was also developed to generate input simulation vectors. For the sake of comparison, we give Table 2 that contains implementation results from [8] and [9], as well as results for proposed architecture equipped with CBSM, or CBSM+.

Table 1 Implementation results and configuration abilities of configurable folded FIR filter with k=3 and N_{MAX}=7

k	N_{MAX}	Clock [ns]	Gate count [KG]	N	k_C	m_C	Reconf [clk]	In.Lat [clk]	Throughput 1/(N·Clock) [MHz]
3	7	7.02	1.78	7	7	3	21	3	20.41
				5	5	3	21	3	28.57
				4	4	3	21	3	35.61

Folded architecture with CBSM, does not provide folding factor reduction, and filters all edges using N_{MAX}=7 filter regardless to the filtering mode. The number of cycles per macro block (MB) required for this architecture is constant (Table 2). The architecture with CBSM+ provides the folding factor reduction, thus the number of filtering cycles depends on video content. Table 2 gives the best, the worst and the number of cycles per MB obtained for well known Foreman video sequence. The best and the worst cycles per MB values are obtained for hypothetical video sequences where all edges are filtered using 4-tap and 7-tap filters, respectively.

Table 2 Comparison of H.264/AVC deblocking filters

	Yuwen Huang's Arch. [8]	Bin Sheng's Arch. [9]	Our arch. with CBSM	Our arch. with CBSM+
Tech.	0.25	0.25	FPGA VirtexE	FPGA VirtexE
Freq. [MHz]	100	100	100	100
Gate count	20.66 K	24.00 K	1.61K **	1.78 K **
Cycles/ MB	614	446	7552	Best - 4480 Worst - 7552 Foreman - 5572
AT*	12685	10704	12158	9918
QCIF	/	/	161.8	181.3
CIF	/	/	40.9	44.8
4CIF	/	/	9.7	11.4
HDTV	45.2	62.3	4.3	5.1

* For Area–Time (AT) measure we take [Gate count]x[Cycl./MB]

** The gate count is obtained as "equivalent gate count" from Xilinx WebPack impl. report. Impl. is performed using VirtexE FPGA

*** For QCIF (176x144), CIF (352x288), 4CIF (704x576), HDTV (1280x720) given values are in [fps].

Table 2 shows that proposed architectures have more than 10 times smaller gate count than architectures proposed in [8] and [9]. Gate count reduction is achieved at cost of time. Our architectures have approximately 10 times greater number of cycles per MB. Proposed architecture with CBSM has nearly the same AT value as the architecture proposed in [8], but 12% worse AT value than [9]. Folded array that exploits reducible folding factor (with CBSM+) has 21% better AT value than [8], and 7.3% better than [9]. It should be noted that configurable folded array with k=3, y=13, N_{MAX}=7 and CBSM can meet the requirement for real-time deblocking of video sequences in format commonly used in mobile devices - CIF (352x288, 30fps) at 74 MHz, while the requirement can be met at 67 MHz with reducible folding factor employment (CBSM+).

6. CONCLUDING REMARKS

In this paper we presented a study of folding technique effectiveness in area-time tradeoff for H.264/AVC deblocking filter implementation. The area-time tradeoffs for configurable folded array were exploited in design of H.264/AVC deblocking filter for embedded mobile computing devices. It resulted in deblocking filter design with extremely low gate count which meets the requirement of real-time deblocking in target applications. The array is restricted for the folded factor at cost of time. In deblocking application, more than 10 times smaller gate count, in respect to designs in [8] and [9], is achieved by nearly 10 time increased number of cycles per MB. The overall product of gate count and number of cycles per MB for the proposed architecture with reducible folding factor is slightly better than [8] and [9]. It makes our platform more efficient for embedded mobile computing applications where computational time, based on image format, is not of primary importance.

REFERENCES

[1] L. Paulson, L. Garber, "Reconfiguring Wireless Phones with Adaptive Chips", *IEEE Computer,* Vol. 36, Number 9, September 2003, pp. 9-11.

[2] I. Richardson, "H.264 and MPEG-4 Video Compression – Video Coding for Next Generation Multimedia", *John Wiley & Sons*, In., New York, 2003.

[3] Z. Yu, J. Zhang, „Video Deblocking with Fine-grained Scalable Complexity for Embedded Mobile Computing", International Conference on Signal Processing, Vol. 2, September 2004, pp. 1173 – 1178.

[4] I. Milentijevic, M. Stojcev, D. Maksimovic, "Configurable Digit - Serial Convolver of Type F", *Microelectronics Journal*, Vol. 27. No. 6, Sep. 1996, pp. 559-566.

[5] I. Milentijevic, V. Ciric, "Assignments of Folding Sets for Adaptive FIR Filtering on Folded Array", Proceedings of the WPS-DSD 2003, 29th Euromicro Conference, Belek, Turkey, September 2003, pp. 21-22.

[6] Y-C. Lin, F-C. Lin, "Classes of Systolic Arrays for Digital Filtering", *Int. J. Electronics*, Vol. 70, No. 4, 1991, pp. 729-737.

[7] K. K. Parhi, "VLSI Digital Signal Processing Systems (Design and Implementation)", *John Wiley & Sons*, In., New York, 2000.

[8] Y. Huang, T. Chen, "Architecture Design for Deblocking Filter in H.264/AVC", *Proceedings of ICME*, Baltimore, Maryland, USA, July 2003, pp. 692-696.

[9] B. Sheng, W. Gao, D. Wu, "An Implemented Architecture of Deblocking Filter for H.Z64/AVC", *International Conference on Image Processing* (ICIP), 2004, pp. 665-668.

[10] V. Ćiric, I. Milentijevic, "Coefficient Bit Reordering Method for Configurable FIR Filtering on Folded Bit-Plane Array", 8th EUROMICRO Conference on Digital System Design, Porto, Portugal, September 2005, pp. 135-138.

[11] JVT software JM10.2, January 2006.