J Sign Process Syst (2017) 89:281–292
DOI 10.1007/s11265-016-1194-6

A Novel Deblocking Filter Architecture for H.264/AVC

Lella Aicha Ayadi[1] · **Taheni Dammak**[1] · **Hassen Loukil**[1] · **Mohamed Ali Benayed**[1] · **Nouri Masmoudi**[1]

Received: 13 October 2014 / Revised: 25 August 2016 / Accepted: 13 October 2016 / Published online: 28 October 2016

Abstract This paper describes efficient hardware architecture for the deblocking filter used in H.264/AVC baseline profile video coding standard. The deblocking filter is a computationally and data intensive tool leading to an increased execution time of both encoding and decoding processes. In fact, we propose a novel edge filter ordering which needs 64 clock cycles to filter a Macroblock (MB). A specified memory organization is also applied in order to avoid unnecessarily waiting for availability of the pixels that will be filtered. The proposed architecture includes both pipelining and parallel processing techniques and is implemented in synthesizable HDL. This hardware is designed to be used as module of a complete H.264/AVC decoder which the functionality was validated on Nios II at 100 MHz.

Keywords H.264/AVC video coding · Deblocking filter · Filter ordering · Hardware implementation

1 Introduction

The video coding standard H.264/AVC [1] or MPEG4 Part 10, developed with the collaboration of ITU-T and ISO/IEC, offers significantly better video compression efficiency than previous video compression standards (such as H.262 and H.263) in terms of bit-rate reduction [1–3]. The performance improved in the coding efficiency is mainly due to incorporation of inter/intra predictions, small block size, integer discrete cosine transform, context-based adaptive binary arithmetic coding (CABAC) and deblocking filter. This latter allows reducing blocking artifacts in a frame due to coarse quantization of MBs and motion compensated prediction. In addition to that, using the deblocking filter, the bit-rate is typically lower by 5 to 10 % and the same objective video quality is preserved [4]. Figure 1 shows the different modules of the H.264/AVC decoder. However, this coding gain is companied with a high computational complexity [5].

In fact, the operation of the deblocking filter is the most complex part of the decoders, consuming one-third of the computational complexity of the H.264/AVC decoder, as shown in Fig. 2. As a consequence of these demanding characteristics, a hardware implementation for such a deblocking filter for high definition video applications is involved, where even larger frame sizes at higher frame rates are to be processed in real-time.

In this paper, we describe the new filtering order and the new memory organization as well as the pipelined and parallel architecture of deblocking filter. The parallelized design is proposed to improve high throughput performing the filtering process of the H.264/AVC decoder in order to achieve real-time performance.

The organization of this paper is as follows. In Sect. 2, we give a brief overview of adaptive deblocking filter algorithm used in H.264/AVC. Sections 3 and 4 present the related works as well as the proposed hardware architecture in details. The simulation results are given in the following section. Section 6 describes the prototype and the performance evaluation. Finally, this paper will be concluded in Sect. 7.

✉ Lella Aicha Ayadi
aicha.ayadi.aa@gmail.com

Nouri Masmoudi
nouri.masmoudi@enis.rnu.tn

[1] Electronics and Information Technology Laboratory, National School of Engineering, University of Sfax, Sfax, Tunisia

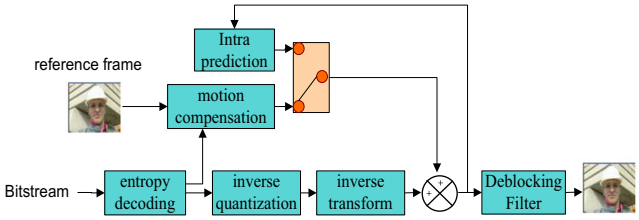


Figure 1 Structure of the H.264/AVC decoder.

2 Deblocking Filter Algorithm

The main objective of the deblocking filter of H.264/AVC is to improve the visual quality of compressed video. In fact, it eliminates visual artifacts generated by the block-based integer discrete cosine transforms (ICTs) in intra and inter frame prediction error coding, coarse quantization of the transform coefficients and motion compensated prediction. In a frame, MBs are filtered in raster scan order. Filtering process is applied to smooth each edge of all the 4 × 4 luma and chroma blocks in a MB except to the edges on the boundary of a frame or a slice [1].

Figure 3 shows the indexes of the 4 × 4 blocks for luminance and chrominance components with the upper neighboring parts (T0 to T7) and the left neighboring blocks (L0 to L7).

The H.264/AVC standard has well defined the filter ordering for all the 4x4 blocks edges in a MB: the vertical edges are filtered from left to right in the order V0…V7 followed by horizontal ones from top to bottom in the order H0…H7. In Fig. 4 the target group of 4 × 4 pixels in a MB (16 × 16 pixels) is indicated by the shadowed area, being filtered in four steps considering the effect of all groups of neighbor pixels . In the first step, the left vertical edge of the block Q0 (called current) is horizontally filtered with data in block P0 (called previous). In the second step, the target group represents a block P1 and the block Q1 is the posterior group. In the third step, the upper horizontal edge of the block Q2 is vertically filtered with the data in block P2. Finally, the horizontal edge separating blocks P3 and Q3 is vertically filtered in the fourth stage.

Nevertheless, according to the position of the 4 × 4 block in a MB to be filtered, the filtering order changes depending on the availability of neighboring blocks as shown in Fig. 5.

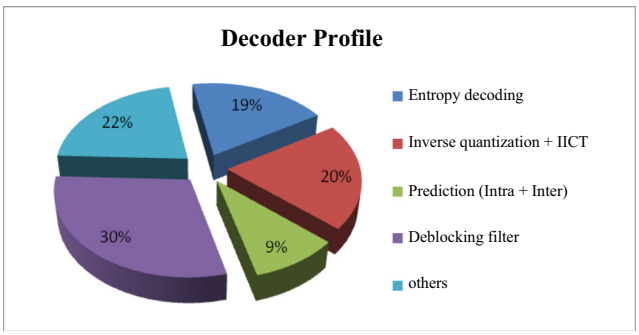


Figure 2 Profiling of H.264/AVC decoder for Foreman video test sequence.

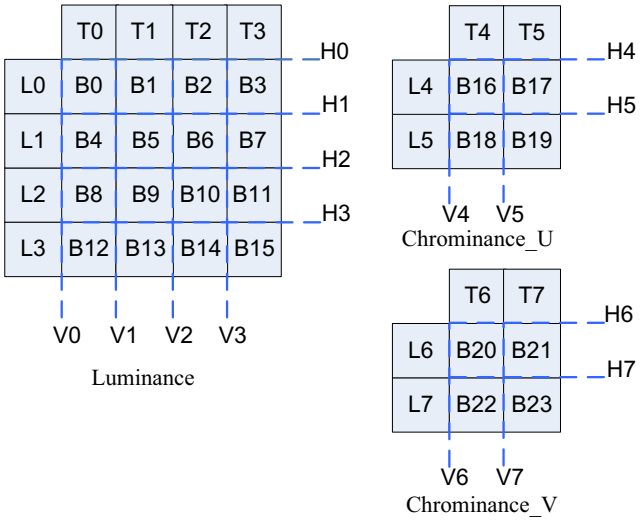


Figure 3 Edges' filtering order of 4x4 luma and chroma blocks in a MB.

For each edge, the appropriate filter is selected from the filter decision according to the boundary strength (BS) which is a number from 0 (no filtering) to 4, thresholds α and β and content of the line of pixels [1]. Horizontal and vertical lines of pixels are illustrated in Fig. 6. For each two neighboring blocks (p and q), the boundary strength (BS) parameter is determined according to Table 1.

For strength 4 (strong filtering) up to three pixels on either side of the edge can be filtered, while for strengths between 1 and 3 (standard filtering) only up to two pixels on either side of the edge may be modified.

When $BS \neq 0$, $|p0 - q0| < \alpha$, $|p1 - p0| < \beta$, $|q1 - q0| < \beta$, the filter will be applied to the desired line of pixels, where the thresholds α and β mainly rely on the quantization parameter (QP) and some other syntax elements [1]. The whole procedure of generating filtered sample values is illustrated in Fig. 7.

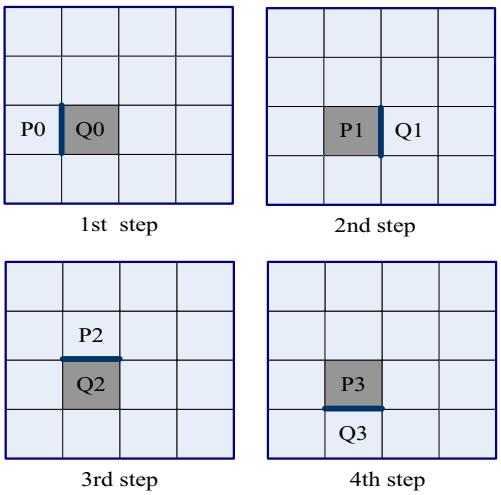


Figure 4 Four stages of the deblocking filter to a 4 × 4 pixels group.

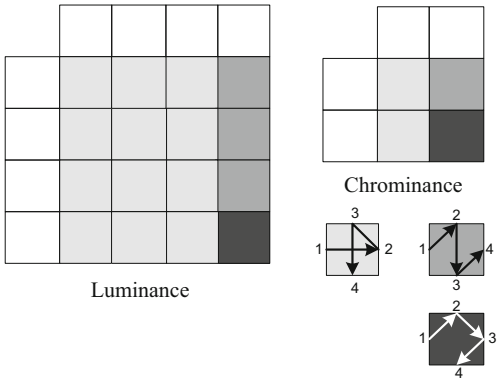


Figure 5 Rules of Edges' filtering order of 4 × 4 block depending on it position in a MB.

3 Related Works

During the last few years, several different hardware accelerators have been presented in the literature for efficient hardware realization of the deblocking filter in H.264/AVC video standard. The processing edge order defined by the H.264/AVC standard is shown in Fig. 8. The vertical edges of the luminance and chrominance blocks are all treated before the horizontal ones.

This sequential order limits the parallel implementation (software or hardware) and the performances of the integrated solution in terms of throughput and latency. Furthermore, this processing order is expensive in terms of memory. Indeed, it generates a large number of temporary data which is 24 4 × 4 blocks (16 in luma block and 4 for each chroma block).

In this case, the idea of G.Khurana [6] was to change the filtering order in order to reduce the local memory size taking on an alternation between the horizontal and vertical filtering of the blocks to be filtered. This order proposed by G.Khurana, shown in Fig. 9, used only a single filter unit to perform all the filtering operations. However, the number of cycles needed to filter 48 edges in a MB is still significantly important.

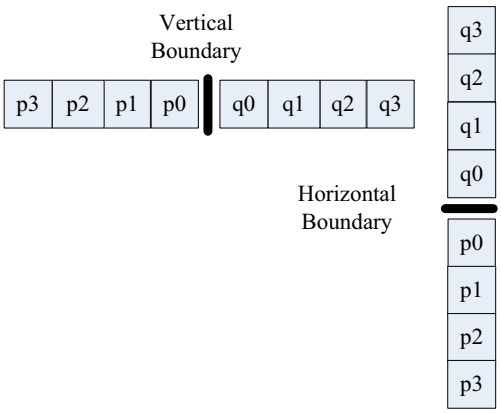


Figure 6 Horizontal and vertical lines of pixels.

Table 1 Decision flow for determining bs values.

Condition	BS
p or q is intra coded and boundary is a macroblock boundary	4
p or q is intra coded and boundary is not a macroblock boundary	3
neither p or q is intra coded; p or q contain coded coefficients	2
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have different reference frames or a different number of reference frames or different motion vector values	1
neither p or q is intra coded; neither p or q contain coded coefficients; p and q have same reference frame and identical motion vectors	0

In similar manner, the proposal of He.Jing [7] is based on multiple filtering units providing less execution time to filter 27 edges than the previous filtering order. This order is represented in Fig. 10.

An efficient architecture that uses six filtering cores was proposed by M.Kthiri [8]. Based on this proposal, a higher level of parallelism was presented in order to meet as possible a minimal number of cycles used to filter only 11 edges in a MB. Nevertheless, this processing order, shown in Fig. 11, increases significantly the use of the local memory. Thus, an additional area is required.

Considering the order in [9], four filters occur in parallel for the luma component but only two filters for the chroma ones as shown in Fig. 12. In order to achieve more performances keeping the same level of parallelism, our solution proposes to exploit four filters for both luma and chroma components.

The design in [10] employs a double deblocking filter with concurrent edge filter and requires 32 BRAMS for data storage.

According to [11], the design adopts three filters for in-loop de-blocking filter (ILF) in H.264/AVC supporting baseline, main, and high profile (BP/MP/HP) video. Nevertheless, the processing rate still significantly important which is around 260 cycles per MB. Although the proposed design in [12] can achieve a processing rate of around 48 cycles per MB, it requires a large storage part consuming 43 % of the total area expense.

In [13], two identical filtering units are proposed allowing on-the-fly filtering of vertical and horizontal edges to increase the filtering performance.

4 Proposed Deblocking Filter Algorithm and Architecture

4.1 Filtering Order

Our filtering order is illustrated in Fig. 13. It provides a better use of the architecture parallelism. Up to four edges are able to be treated at the same time. The repeated numbers correspond

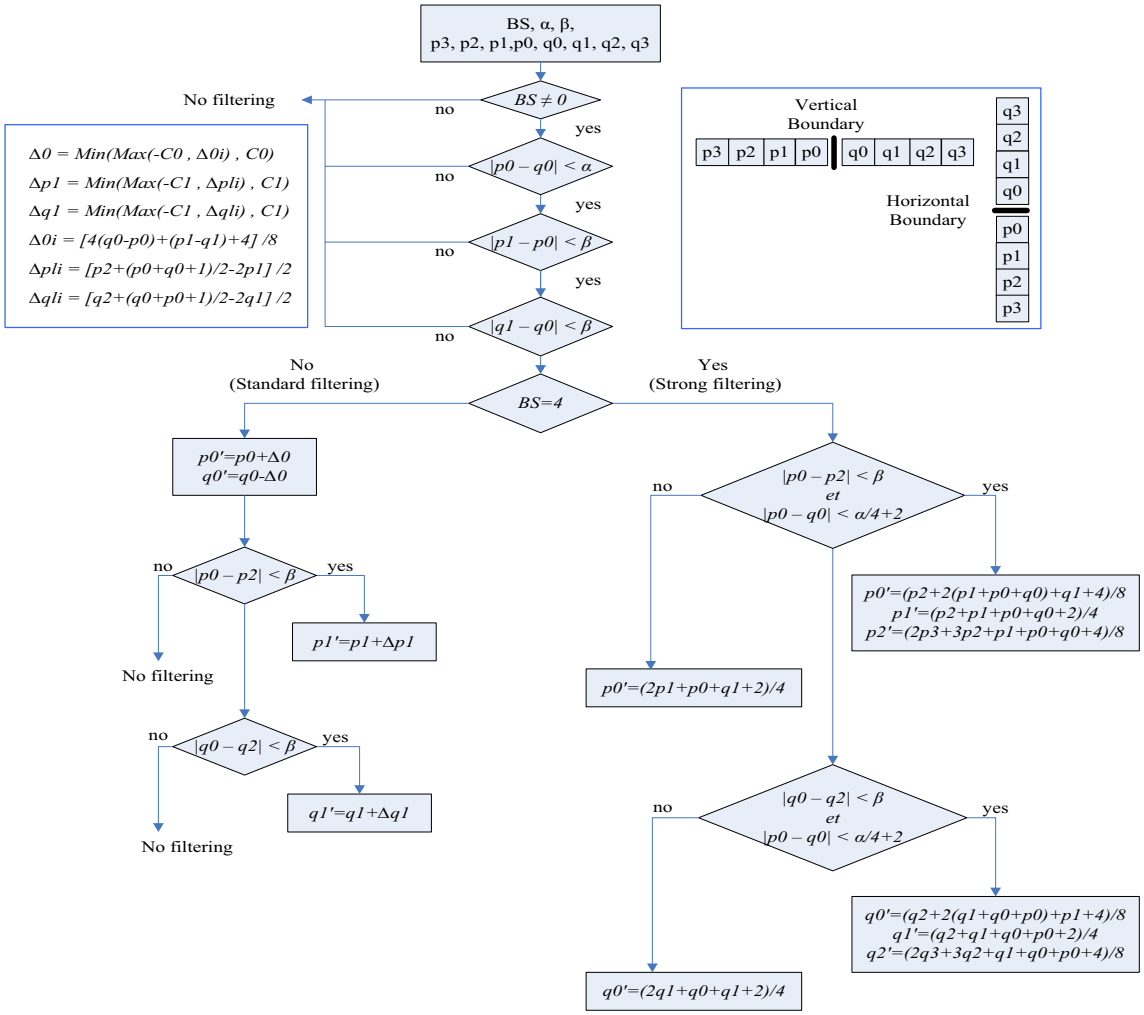


Figure 7 Deblocking filter flowchart.

to the filtering which occurs in parallel. As a result, only 12 edges have to be filtered in a MB.

Considering this order, four filtering occur in parallel producing the same result as specified by the H.264/AVC standard.

4.2 Filter Architecture

The internal architecture of the deblocking filter is shown in Fig. 14. The designed architecture was described in VHDL. Once data and associated parameters are ready to be learned, the process of filtering starts to generate the filtered MB that

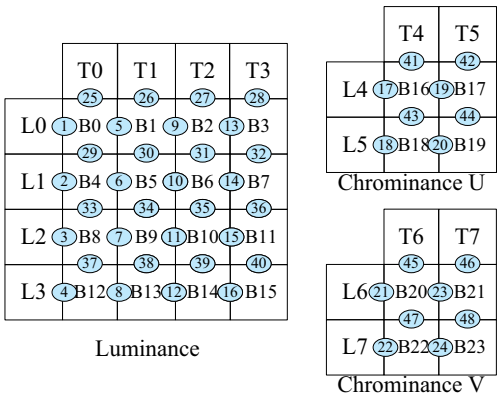


Figure 8 Original H.264/AVC filtering order.

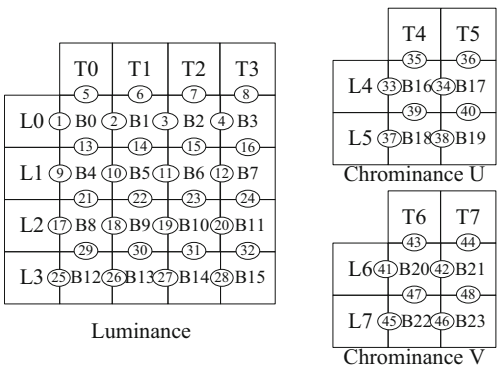


Figure 9 Filtering order proposed by G.Kurana [6].

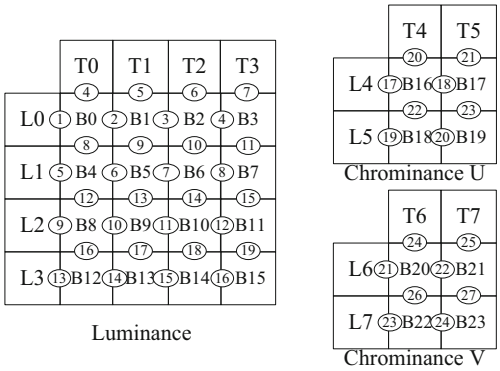


Figure 10 Filtering order proposed by He.Jing [7].

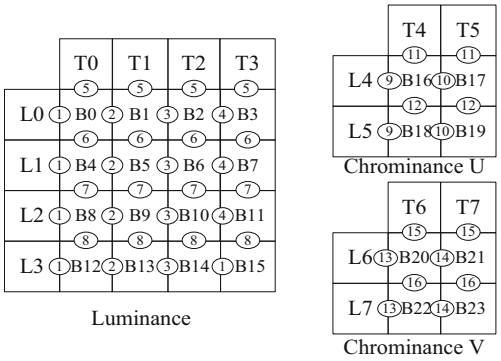


Figure 12 Filtering order proposed by H.Loukil [9].

will be later stored in the appropriate memory. All modules of the internal architecture are synchronized with a control unit.

The role of the "Input Data Store" block, shown in Fig. 14, is to concatenate on 128 bits the sixteen pixels to be filtered; corresponding to a 4 × 4 block.

Figure 15 shows our proposed memory organization. In order to guarantee one clock cycle transfers, we propose to use eight 5 × 128-bit memory modules. The data storing in the appropriate memory is synchronized with a "Control Unit" by control signals. In fact, these memories are controlled by the signals « wren » and « rden ». If wren = '1', the memory receives data and stores them into appropriate addresses. Else if rden = '1', we can read then from the memory the different stored pixels by giving only the desired pixel address provided by the "Control Unit". Before starting the filtering process, all data have to be stored in the appropriate memory modules in 47 cycles. Indeed, 28 cycles are required to store the luma block values and the neighboring pixels left and right. Then 19 cycles are taken to store the pixel of chroma blocks and neighborhoods. The process of data storage is represented in Fig. 16.

The "Calculator Unit" module allows computing indexA and indexB that are defined as clipping values which essentially depend on the quantization parameter.

The "Filter unit" module uses one calculator module to define some parameters such as c1 and threshold values (α and β) by referring to tables given by the H264/AVC standard.

Furthermore, this block uses sixteen filter 1-D units in order to process in parallel and simultaneously the filtering of four edges. Each one allows the filtering of a line of pixels. The internal filter 1-D architecture is represented by Fig. 17. The filtering process performs the filtering operations using the pixels not filtered and the values of BS, thresholds (α and β) and c1 value, which were previously calculated. The "Parameters calculator", shown in Fig. 17, generates the filtering parameters used in the deblocking filter algorithm.

The FIFO 1, 2, 3 and 4, represented in Fig. 14, allows storing the intermediate results of the horizontal filtering to make the vertical filtering later. The organization of these memories is illustrated in Fig. 18. FIFO 1 and 2 are two 5 × 128-bit FIFO memories while FIFO 3 and 4 are two 4 × 128-bit FIFO memories.

Because the proposed architecture is designed to perform both horizontal and vertical filtering of four 4 × 4 block edges using the same filter, pixels horizontally filtered in each block must be transposed after being storing in their appropriate memory and before starting vertical filtering. In fact, the "Transpose Unit" is required to transpose a 4 × 4 block of pixels from rows to columns as Fig. 19 illustrates. The

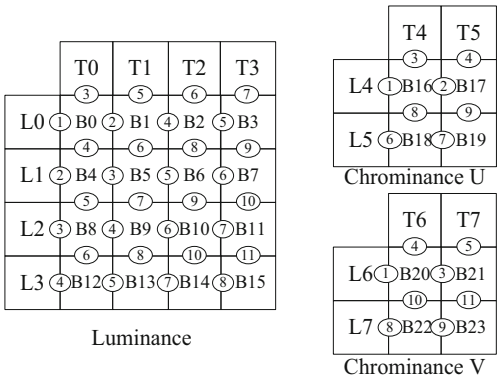


Figure 11 Filtering order proposed by M.Kthiri [8].

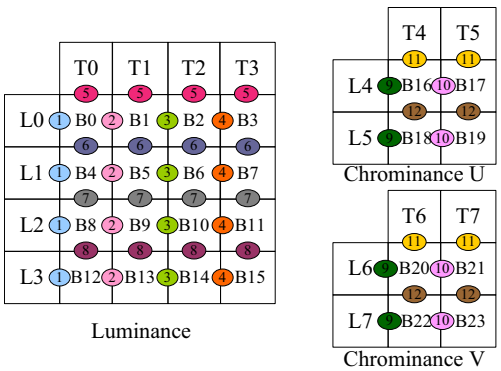


Figure 13 Proposed filtering order.

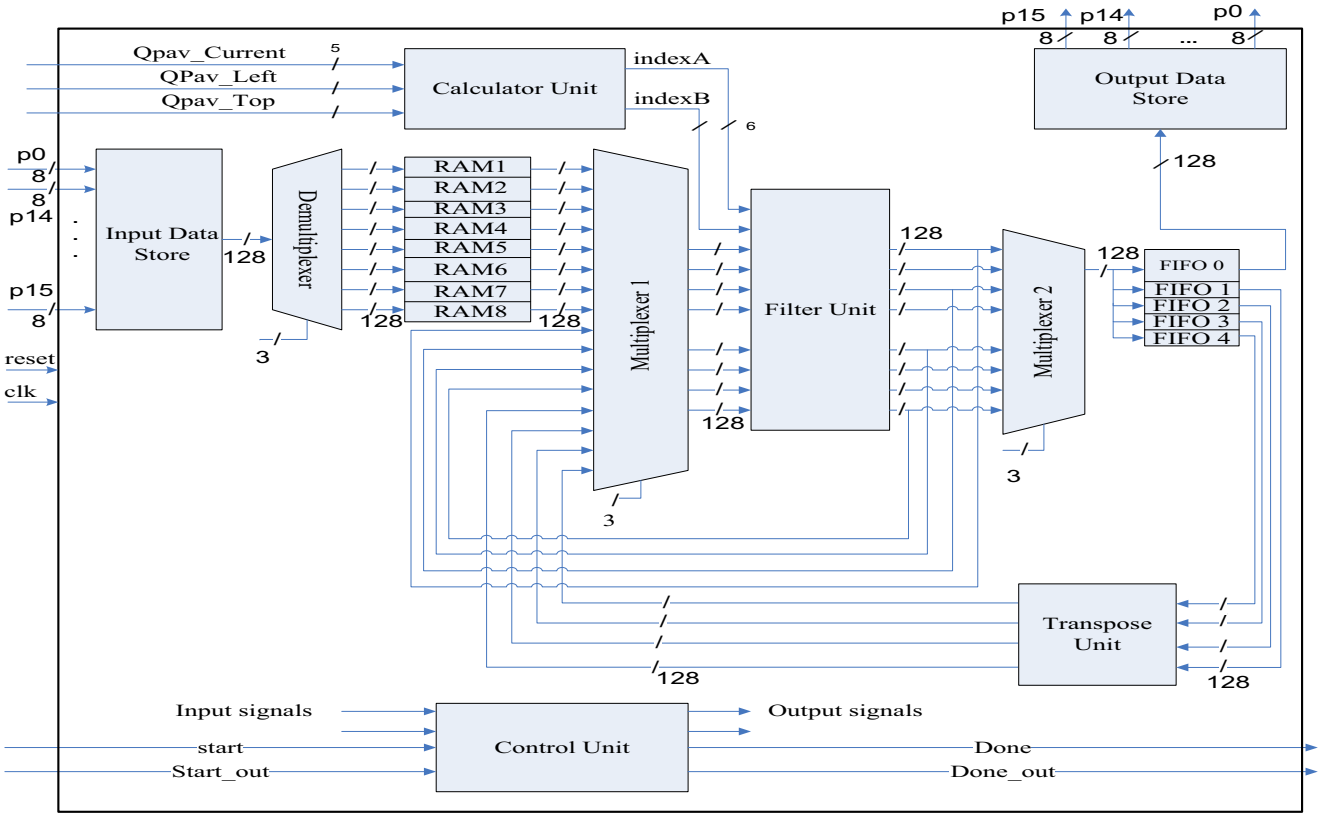


Figure 14 Proposed deblocking filter architecture.

implemented "Transpose Unit" completes the transposition of a 4 × 4 block in 1 clock cycle. The MB totally filtered will be stored in FIFO 0 (40 × 128-bit FIFO memory). To have the result of filtering, we get to empty FIFO 0 in 40 clock cycles. For each clock cycle, the separation between the pixels constituting a 4 × 4 block is performed by the "Output Data Store".

The proposed architecture operates in a parallel structure that performs four filtering simultaneously. The process of filtering is schematized with the timing diagram presented in Fig. 20 where Bt represents the transposed block. Therefore, the first 3 clock cycles are needed to filter pixels that are associated with the four initial edges (assigned 1). In fact, the blocks on either side of this edge (L0, B0, L1, B4, L2,

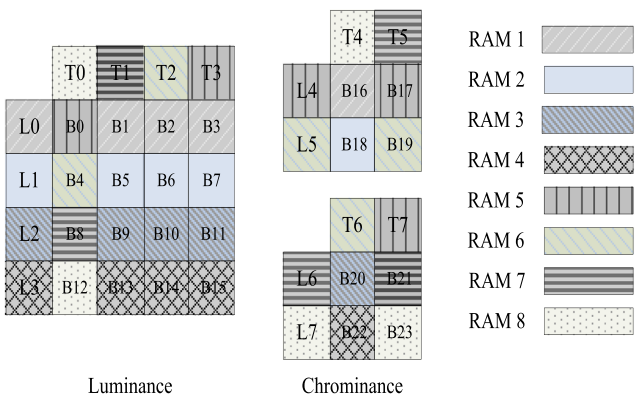


Figure 15 Memory organization.

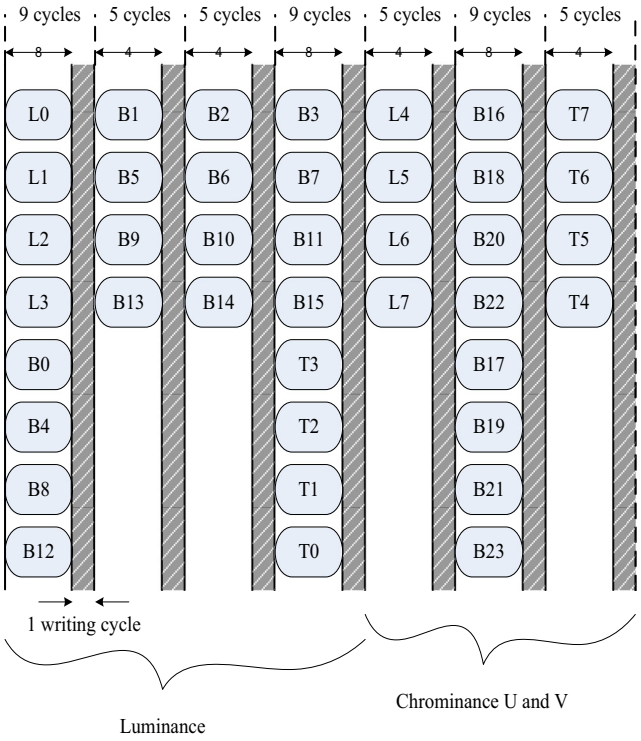


Figure 16 Data storage.

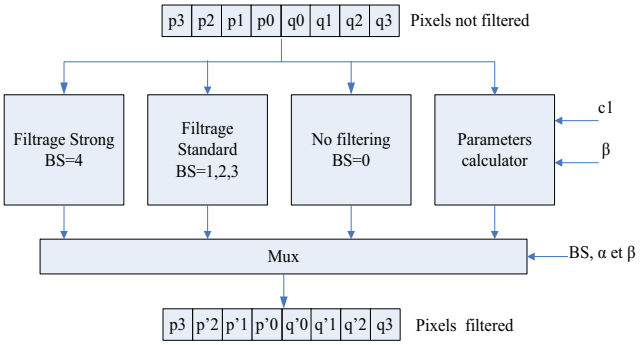


Figure 17 Architecture of filter 1-D.

B8, B12 and L3) are respectively reached from the following memories: RAM 1, RAM 5, RAM 6, RAM 2, RAM 3, RAM 7, RAM 4 and RAM 5, by taking just one read cycle. After a clock cycle reserved for the multiplexer 1, these data are ready to be filtered where the filtering takes a clock cycle too. Then, the blocks L'0, L'1, L'2 and L'3 are forwarded to FIFO 0 to be stored in four clock cycles. While these blocks are storing, the blocks B'0, B'4, B'8 and B'12, partially filtered, are directly transmitted to the appropriate filters for the next edge. During the next four clock cycles, the blocks B"0, B"4, B"8 and B"12 (horizontally filtered) are transferred to the temporary memory FIFO 1 and the filtering of the third edge is performed.

The horizontally filtered blocks B1", B5", B9" and B13" will be stored in FIFO 2. The filtering of vertical edges of the Luma component is well explained by Fig. 21. Regarding the vertical filtering, horizontal edge needs 5 clock cycles to be filtered. Indeed, unlike the horizontal filtering, one extra clock cycle will be reserved to transposition as shown in Fig. 22. After achieving the filtering of the edges assigned 5 (T0|B0, T1|B1, T2|B2, T3|B3), the blocks T0', T1', T2'and T3' totally filtered will be stored in FIFO 0 in 4 clock cycles. After 42 clock cycles, the entire Luma component is filtered and stored in FIFO 0. As for the filtering of Chroma edges, it will be similar to Luma ones. Thus, the proposed design is able to finalize the whole filtering of a MB in 64 cycles.

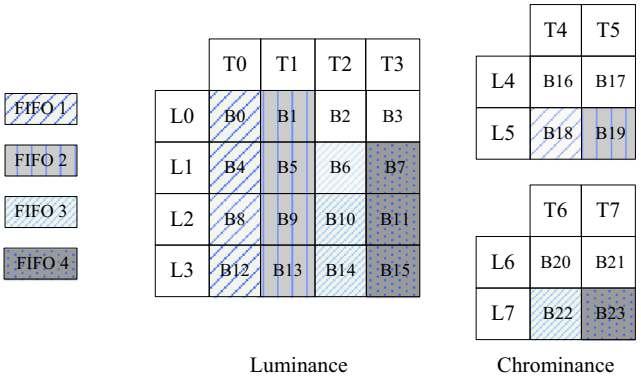


Figure 18 Temporary memory organization.

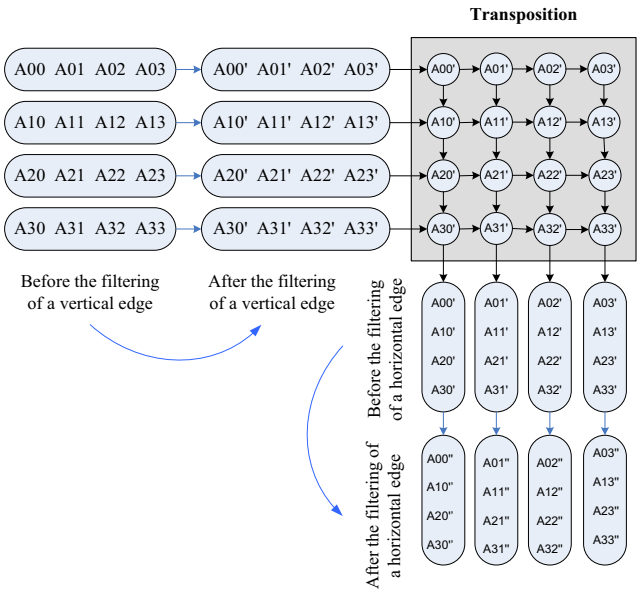


Figure 19 Transpose unit.

5 Simulation Results

In order to verify the compatibility with the H.264/AVC standard, the exactitude of the implemented architecture has been checked with RTL simulations using ModelSim ALTERA. Then, the VHDL code is synthesized to an ALTERA Stratix III FPGA using ALTERA Quartus. As explained before, the architecture proposed in this paper considered a new filter ordering allowing the whole filtering of a MB in 64 cycles. 47 cycles are reserved for storing data in the appropriate memories. For output data, 40 cycles are needed to have a filtered MB. Deblocking filter synthesis results are represented in Table 2. Figure 23 represents the area profiling of the proposed design in terms of ALUTs, logic registers and memory bits.

Furthermore, the Table 3 shows the performance of our hardware architecture among various related works. This work uses the same level of parallelism as previous work in [9], but processes twice times higher throughput. It achieves 2343 kMB per second which is sufficient to process 4 K (4096 × 2160 @ 60fps) application. This hardware design also achieves higher performance than the hardware architecture presented in [10, 11, 13] in terms of throughput. Table 3 highlights that the designs in [8, 12] are competitive in hardware efficiency when compared with the proposed design in term of processing time. However, [8, 12] require large local memory that results in additional area cost. Indeed, design in [12] can process one MB data in 48 cycles at a cost of 41.6 K gates along with 640 bytes single-port SRAM. Deblocking filter hardware accelerator in [8] uses six filter units (three filters for the horizontal edges and three filters for the vertical edges). The rest of this architecture is composed by fourteen local memory modules, a BS calculator unit, one threshold

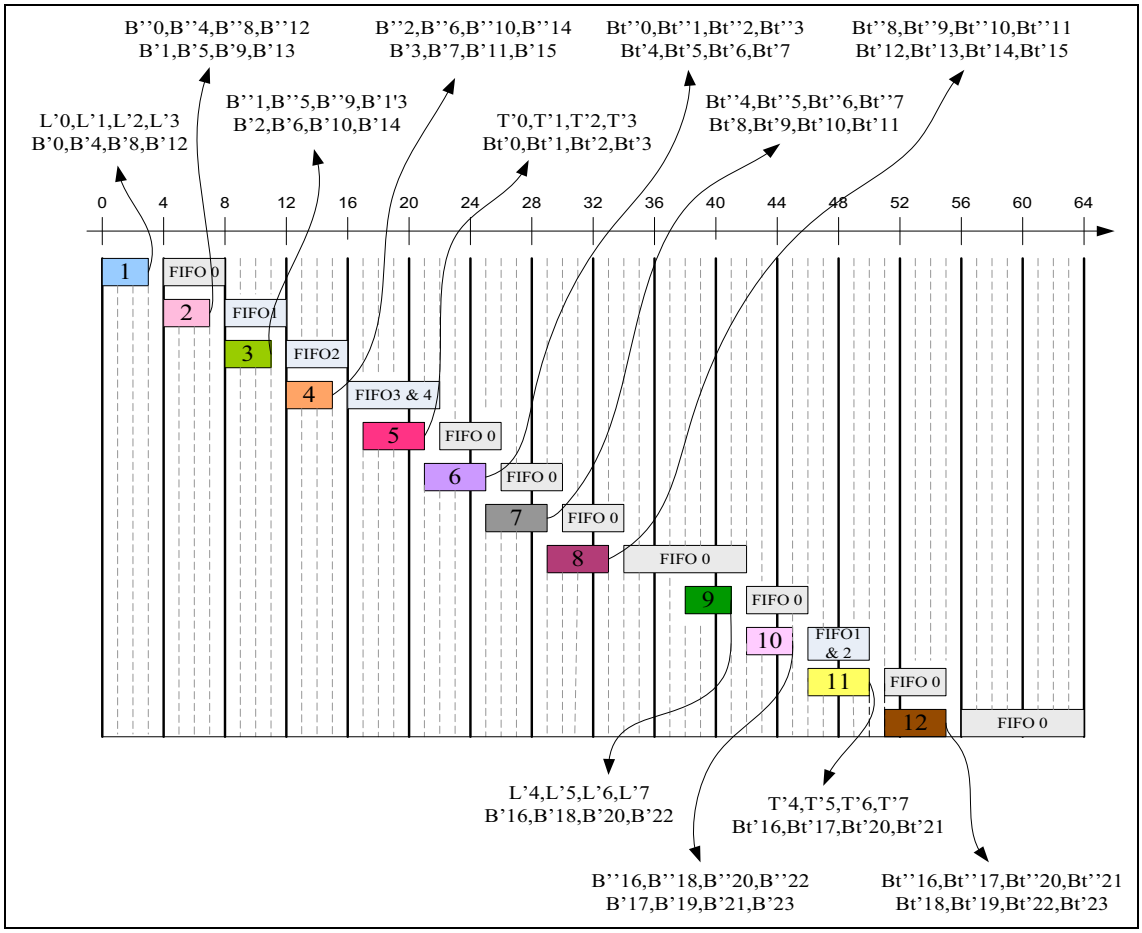


Figure 20 Timing diagram of the proposed architecture.

calculator module, one c1 calculator module, twelve transpose modules, six 4 × 4 register arrays and thirteen 4 × 32-bit temporal buffers. This architecture consumes 11,830 LUTs considering only the area of six filer units, as shown in Table 3.

For fair area comparison with [8], we performed implementation of our architecture on Xilinx Virtex V FPGA device. If we exclude the rest of our design, it requires only 10,121 LUTs. Therefore, the hardware cost of [8] increases in term of the final area of the architecture. In summary, the proposed design can be beneficial for small-sized FPGAs achieving high throughput due to low processing cycles and relative high frequency.

6 Prototype and Performance Evaluation

6.1 Prototype

To validate the functionality of the proposed hardware architecture of the deblocking filter algorithm in a practical realization, a video decoder was developed and implemented on Altera Stratix III development platform making use of a Stratix III EP3SL150 FPGA device. In addition to all the implementation resources offered by the reconfigurable device, this platform also can uses the Altera NIOS II softcore processor and provides several block RAM modules and high

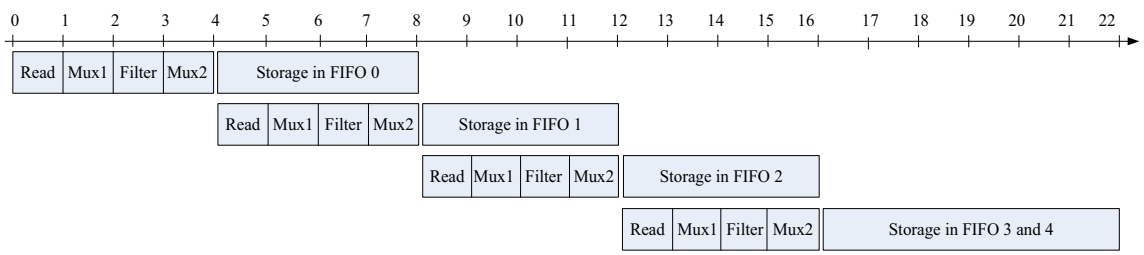


Figure 21 Filtering of vertical edges in multiple parallel pipeline.

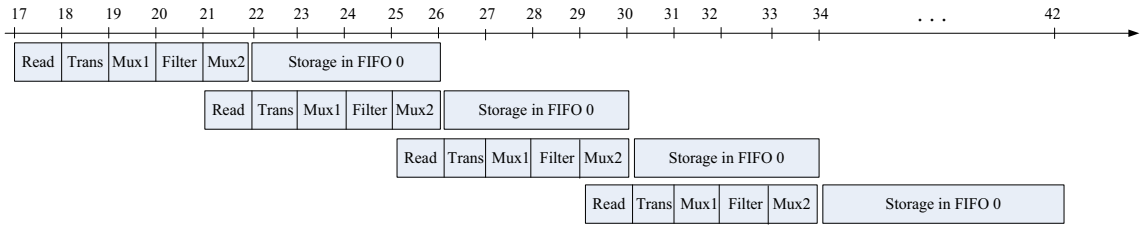


Figure 22 Filtering of horizontal edges in multiple parallel pipeline.

speed on-chip bus communication links. The main avail of this softcore processor is its extensibility and adaptability to incorporate custom logic directly into the NIOS II arithmetic logic unit (ALU). Furthermore, the interconnection between the NIOS II and several peripherals is implemented by a dedicated bus (Avalon Bus). Altera introduces the SOPC builder tool, for the quick creation and easy evaluation of embedded systems. The integration off-the-shelf intellectual property (IP) as well as reusable custom components is realized in a friendly way, reducing the required time to set up a SoPC (system on programmable chip).

6.2 Integration with the Embedded Video System

Architecture validation was realized by integrating the deblocking filter algorithm as a coprocessor into the embedded video decoding system which the components are illustrated in Fig. 24. The proposed embedded system consists of three major parts, including the NIOS II softcore processor, the deblocking filter coprocessor and the peripheral interface modules. All these modules are connected to the 32-bits Avalon bus. The deblocking filter coprocessor uses two buses to input/output data which are *data_in* and *data_out* as shown in Fig. 24. The bus widths are 32 bits.

The Altera NIOS II softcore processor (FAST version) is configured as follows: a 32-bits scalar RISC processor with Harvard architecture, 6 stage pipeline, 1-way direct-mapped 64 KB data cache, 1-way direct-mapped 64 KB instruction cache and can gives up to 200 MIPS. Such interconnect buses are used not only to exchange the control signals between the NIOS II and the deblocking filter coprocessor, but also to send all the required data to filtering process.

The NIOS II processor executes a software program that is loaded into the DDR memory. This software is written in C language and is used to check if the deblocking filter coprocessor is not busy with the *waitrequest* signal. In this case, our coprocessor loads the current MB and its neighboring blocks through the 32-bits *data_in* signal and activates the data processing. During the calculation step, the coprocessor is busy and cannot be accessed.

At the end of processing, the *waitrequest* signal has a low level state and the coprocessor provides the filtered pixels through the 32-bits *data_out* signal. Indeed, in the purpose of using the 32-bits bus size, each four 8-bits coefficient must be processed as a 32-bits long word in order to decrease the memory access.

Figure 25 presents an example to read/write data from/to deblocking filter coprocessor. In fact, the deblocking filter coprocessor receives *address*, *read* or *write* and *writedata* signals after the rising edge of the clock. The deblocking filter coprocessor asserts if necessary *waitrequest* before the next rising clock edge to hold off the transfers. When the coprocessor asserts *waitrequest*, the transfers are delayed and the address and control signals are held constant. The transfers are completed on the rising edge for the first *clk* after the coprocessor reasserts *waitrequest*.

Table 2 Deblocking filter synthesis results.

ALUTS	15.978/113.600 (14 %)
Input/output	292/744
Registers	17.253/113.600 (15 %)
Memory bits	12.288/5.630.976 (1 %)

6.3 Performance Evaluation

The embedded video system is used to verify and evaluate the performances of our deblocking filter coprocessor in HW/SW (hardware/software) context. Indeed, it is designed for accelerating computation for the H.264/AVC decoder. The processor core clock and system clock are set to 100 MHz. Taking into account the performances of the embedded video system using the Altera Stratix III development platform, we have measured the execution times of the deblocking filter in SW and HW/SW by using the NIOS II timer "high_res_timer" which can be used for the cycle-accurate time-frame estimation. Once the whole design are described in VHDL at the

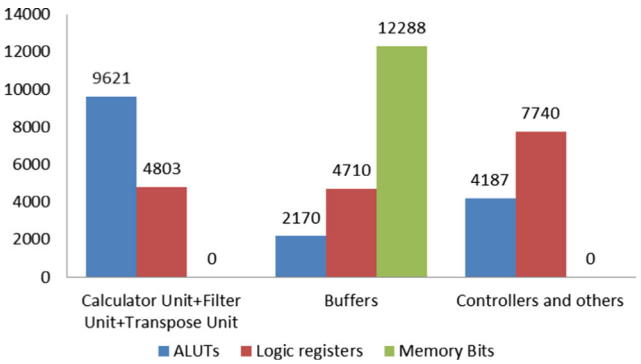


Figure 23 Hardware complexity profiling.

Table 3 Comparisons among related works.

Algorithm	[8]	[9]	[10]	[11]	[12]	[13]	Proposed
Technology	VirtexV	StratixII	VirtexV	0.13 µm	0.18 µm	0.18 µm	StratixIII
Level of parallelism	6	4	2	3	4	2	4
ALUTs or Gates	11,830[b]	23,874	16,594	36,900	41,600	12,600	15,978
Processing time (cycles/MB)	49	105	75	260	48	110	64
Working Frequency (MHz)	123.75	150	43.4	225	135	100	150
Throughput[a] (kMB/s)	2525	1428	578	865	2812	909	2343

[a] Throughput (kMB/s) = Frequency (kHz) /processing time (cycles/MB)

[b] Area considering only six filter units

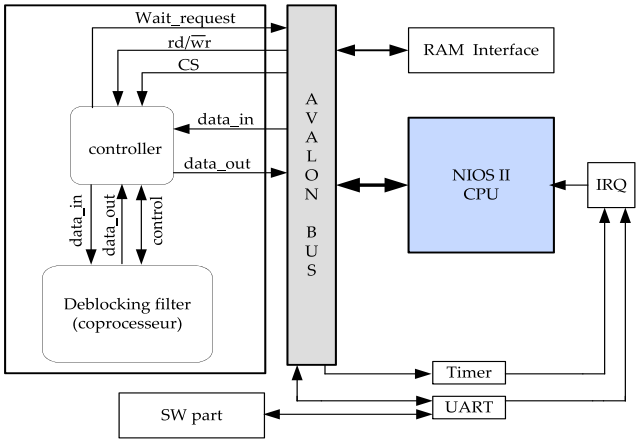


Figure 24 Embedded video system.

RTL level and fitted into the FPGA, we have determined decoding time of the process of filtering. Table 4 shows a comparison of the processing time (ms) necessary to decode 100 frames of different video test sequences (CIF 352 × 288 @ 30 fps) with the deblocking filter by SW and HW/SW solutions. We focus on the following standard video test sequences: “News”, “Foreman”, and “Akiyo” having different movements and camera operations. From these results, we can conclude that the gain is estimated to 20 % of processing time. Although this order can be considerable, it will still be limited by the significant transfer data between the deblocking filter coprocessor and SDRAM memory.

7 Conclusion

In this paper, we proposed an efficient hardware architecture operating four filters simultaneously for real-time implementation of H.264 adaptive deblocking filter algorithm and generating a filtered MB in 64 clock cycles. Also, we have presented a modern implementation of the complex video application such as H.264/AVC decoder in HW/SW solution.

In fact, the deblocking filter component has been integrated as a coprocessor into embedded video system for improving the system performances. The deblocking filter in HEVC relies on the same principles as in H.264/AVC but it differs in ways that have a significant impact on complexity. Indeed, HEVC limits the filtering to the edges lying on an 8x8 grid so that reduces by half the number of filter modes and the number of samples that may be filtered. However, there are also aspects of HEVC that increase the complexity of the filter such as the addition of clipping in the strong filter mode. In other words, this work may be modified in order to support the filtering process in HEVC.

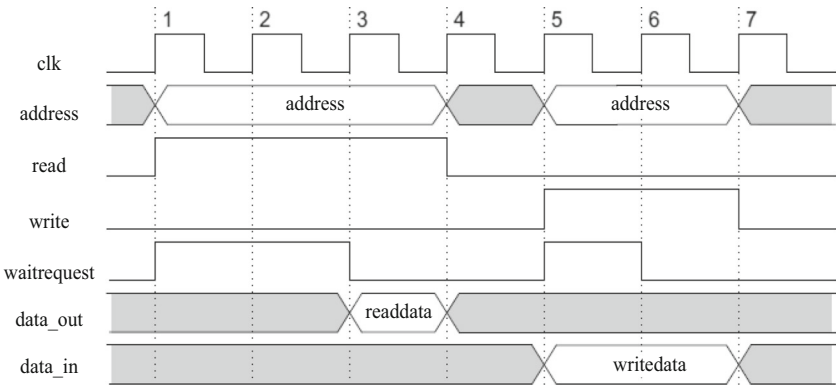


Figure 25 Read/write data from/to deblocking filter coprocessor.

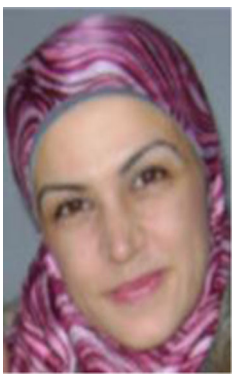
Table 4 Processing time (ms) of the deblocking filter algorithm with sw and hw/sw solutions.

	News	Foreman	Akiyo
SW	743.57 ms	735.04 ms	523.08 ms
HW/SW	560.2 ms	608.81 ms	460.31 ms
Gain (%)	24.66	17.17	17.2

References

1. Joint Video Team of IT-T VEG and ISO/IEC MPEG. (2003), "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification", ITU-T Rec. H.264 and ISO/IEC 14496–10 AVC.
2. Sullivan, G. Topiwala, P & Luthra, A. (2004). The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. *SPIE Conf. On Apps. Of digital Image Processing* (pp. 454–474).
3. Ostermann, J., Bormans, J. P., List, P., Maroe, D., Narroschke, M. F., Pereira, F., et al. (2004). Video coding with H.264/AVC: tools, performance and complexity. *IEEE Circuit and Systems Magazine, 4*(1), 7–28.
4. Wiegand, T., Sullivan, G. J., Bjontegaard, G., & Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(7), 560–576.
5. Horowitz, M., Joch, A., Kossentini, F., & Hallapuro, A. H. (2003). 264/AVC baseline profile decoder complexity analysis. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(7), 704–716.
6. Khurana, G., Kassim, T., Chua, T., & Mi, M. (2006). A pipelined hardware implementation of in-loop deblocking filter in H.264/AVC. *IEEE Transactions on Consumer Electronics, 52*(2), 536–540.
7. Jing, H. Yan, H. Xinyu, X. (2009). An Efficient Architecture for Deblocking Filter in H.264/AVC. *International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (pp. 848–851).
8. Kthiri, M. Kadionik, P. Lévi, H. Loukil, H. Ben Atitallah, A. Masmoudi, N. (2010). A Parallel Hardware Architecture of Deblocking Filter in H264/AVC. *International Symposium on, Electronics and Telecommunications (ISETC) (pp. 341–344*).
9. Loukil, H. Ben Atitallah, A. Masmoudi, N. (2009). Hardware architecture for H.264/AVC deblocking filter algorithm. *IEEE International Conference on Systems, Signals and Devices* (pp 1–6).
10. Ernst, E. (2007). Architecture Design of a Scalable Adaptive Deblocking Filter for H.264/AVC, New York, MSc Dissertation, Rochester.
11. Chien, C.A., Chang, H.C., Gue, J.I. (November 30 - December 3 2008). A high throughput in-loop de-blocking filter supporting H.264/AVC BP/MP/HP video coding. In *Proceedings of the IEEE Asia Pasific Conference on Circuits and Systems (APCCAS'08)* (pp. 312–315).
12. Chen, K. H. (2010). 48 cycles-per-macro block deblocking filter accelerator for high-resolution H.264/AVC decoding. *IET Circuits, Devices and Systems, 4*(3), 196–206.
13. Tobajas, F., CalIicó, G. M., Perez, P. A., de Armas, V., & Sarmiento, R. (2008). An efficient double-filter hardware architecture for H.264/AVC deblocking filtering. *IEEE Transactions on Consumer Electronics, 54*(1), 131–139.

Lella Aicha Ayadi was born in Sfax, Tunisia, in 1989. She received her engineer diploma in 2013 from the National Engineering School of Sfax (ENIS), Tunisia. Currently, she is working toward her Ph.D. degree in Electronic Engineering at the Laboratory of Electronics and Information Technology (LETI)-ENIS. Her main research activities are focused on image and video signal processing, hardware implementation and embedded systems. She is a member of IEEE.



Taheni Dammak Received electrical engineering degree from the National School of Engineering-Sfax (ENIS) in 2006. She received his M.S. and Ph.D. degrees in electronics engineering from Sfax National School of Engineering in 2007 and 2013 respectively. She is currently an assistant professor at Higher Institute of Electronic and Communication of Sfax (Tunisia). She is teaching courses and tutorials of signal processing for license level in telecommunication. She is currently researcher in the Laboratory of Electronics and Information Technology and an assistant at the University of Sfax, Tunisia. Her main research activities are focused on image and video signal processing, DSP (Digital Signal Processing), embedded systems and C programing langage.

Hassen Loukil Received electrical engineering degree from the National School of Engineering-Sfax (ENIS) in 2004. He received his M.S. and Ph.D. degrees in electronics engineering from Sfax National School of Engineering in 2005 and 2011 respectively. He is currently an assistant professor at Higher Institute of Electronic and Communication of Sfax (Tunisia). He is teaching Embedded System conception and System on Chip. He is currently researcher in the Laboratory of Electronics and Information Technology and an assistant at the University of Sfax, Tunisia. His main research activities are focused on image and video signal processing, hardware implementation and embedded systems.

Mohamed Ali Ben Ayed was born in Sfax, Tunisia, in 1966. He received his B.S. degree in Computer Engineering from Oregon State University and M.S. degree in Electrical Engineering from Georgia Institute of Technology in 1988, his DEA, Ph.D., and HDR degrees in Electronics Engineering from Sfax National School of Engineering in 1998, 2004, and 2008, respectively. He is currently a professor in the Department of Communication at Sfax High Institute of Electronics and Communication. He was a co-founder of "Ubvideo Tunisia" in the techno-pole El- GHAZLA Tunis, an international leader company in the domain of video coding technology. He is a member of a research team since 1994 at (LETI, Sfax) in the domain of Electronics and Information Technology and a reviewer in many international and national journals and conferences. He is currently a technical advisor of "EBREASK video", a Research and Development company specialized on the next high efficient video coding generation H265. His current research interests include DSP and VHDL implementation of digital algorithms for multimedia services, and development of digital video compression algorithms.

Nouri Masmoudi received his electrical engineering degree from the Faculty of Sciences and Techniques—Sfax, Tunisia, in 1982, the DEA degree from the National Institute of Applied Sciences—Lyon and University Claude Bernard—Lyon, France in 1984. From 1986 to 1990. He received his Ph.D. degree from the National School Engineering of Tunis (ENIT), Tunisia in 1990. He is currently a professor at the electrical engineering department—ENIS. Since 2000, he has been a group leader 'Circuits and Systems' in the Laboratory of Electronics and Information Technology. Since 2003, he is responsible for the Electronic Master Program at ENIS. His research activities have been devoted to several topics: Design, Telecommunication, Embedded Systems, Information Technology, Video Coding and Image Processing