Motion Compensation Hardware Accelerator Architecture for H.264/AVC

Bruno Zatt[1], Valter Ferreira[1], Luciano Agostini[2], Flávio R. Wagner[1], Altamiro Susin[3], and Sergio Bampi[1]

[1] Informatics Institute
Federal University of Rio Grande do Sul
Porto Alegre – RS – Brazil
[2] Informatics Department
Federal University of Pelotas
Pelotas – RS – Brazil
[3] Electrical Engineering Department
Federal University of Rio Grande do Sul
Porto Alegre – RS – Brazil
{bzatt, vaferreira, flavio, bampi}@inf.ufrgs.br
agostini@ufpel.edu.br
Altamiro.Susin@ufrgs.br

Abstract. This work presents a new hardware acceleration solution for the H.264/AVC motion compensation process. A novel architecture is proposed to precede the luminance interpolation task, which responds by the highest computational complexity in the motion compensator. The accelerator module was integrated into the VHDL description of the MIPS Plasma processor, and its validation was accomplished by simulation. A performance comparison was made between a software implementation and a hardware accelerated one. This comparison indicates a reduction of 94% in processing time. The obtained throughput is enough to reach real time when decoding H.264/AVC Baseline Profile motion compensation for luminance at Level 3.

Keywords: Video Coding, H.264/AVC, MPEG-4 AVC, Motion Compensation, Hardware Acceleration.

1 Introduction

Currently, the development of embedded devices that use some system of video player is growing. Such systems need to find a balance between the computational complexity, to execute their functions, and the excessive increasing in the energy consumption.

On the other hand, the H.264/AVC standard of video [1,2] compression, due to its high complexity, needs powerful processors and hardware support to work accordingly with the application requirements. Furthermore, the motion compensation operation presents one of the highest computational complexities in a H.264/AVC [3] decoder. This high complexity also implies in large energy consumption. This work intends to generate an efficient embedded solution for the H.264/AVC motion compensation.

D. Mery and L. Rueda (Eds.): PSIVT 2007, LNCS 4872, pp. 24 – 35, 2007.

In this work, a general purpose processor was used together with a specific designed accelerator hardware to meet the embedded motion compensation requirements. The processor used was the MIPS Plasma processor, and the two-dimensional FIR filter was designed as the accelerator hardware. Then a satisfactory real time performance was obtained for the motion compensation process. As the operation frequency of Plasma is relatively low (74MHz), the energy consumption of this solution could be lower than that obtained through the design of the complete motion compensation in hardware. Other advantage is the time-to-market, once processor-based systems are more quickly designed than specific integrated circuits.

This paper is organized as follows. Section 2 presents the H.264/AVC standard. The motion compensation process in the H.264/AVC and its main features are presented in the third section. In Section 4, the proposed MC hardware accelerator architecture is presented in details. The integration with the MIPS processor is shown in Section 5. Section 6 presents the synthesis results and the performance comparison. Finally, Section 7 concludes the work.

2 The H.264/AVC Standard

H.264/AVC [1] is the latest video coding standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). H.264/AVC provides higher compression rates than earlier standards as MPEG-2, H.263, and MPEG-4 part 2 [2].

The H.264/AVC decoder uses a structure similar to that used in the previous standards, but each module of a H.264/AVC decoder presents many innovations when compared with previous standards as MPEG-2 (also called H.262 [4]) or MPEG-4 part 2 [5]. Fig. 1 shows the schematic of the decoder with its main modules. The input bit stream first passes through the entropy decoding. The next steps are the inverse quantization and inverse transforms (Q-1 and T-1 modules in Fig. 1) to recompose the prediction residues. Motion compensation - MC (also called INTER prediction) reconstructs the macroblock (MB) from neighbor reference frames, while INTRA prediction reconstructs the macroblock from the neighbor macroblocks in the same frame. INTER or INTRA prediction reconstructed macroblocks are added to the residues, and the results of this addition are sent to the deblocking filter. Finally, the reconstructed frame is filtered by the deblocking filter, and the result is sent to the frame memory. This work focuses on the motion compensation module, which is highlighted in Fig. 1.

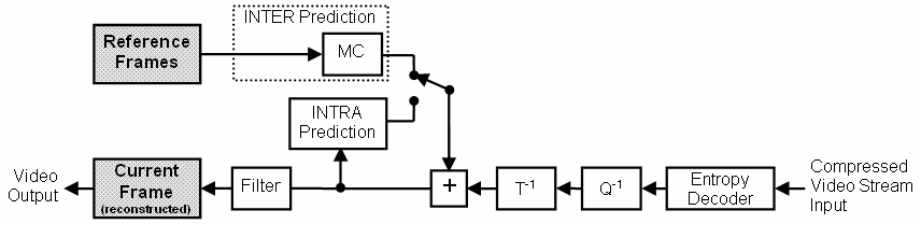

Fig. 1. H.264/AVC decoder diagram

H.264/AVC was standardized in 2003 [1] and defines four profiles, targeting different applications. These profiles are called: Baseline, Main, Extended, and High. The Baseline profile (which is the focus of this work) focuses on low delay applications and was developed to run on low-power platforms. The Main profile is oriented to high image quality and HDTV applications. It added some different features with regard to the Baseline profile, like: bi-prediction, weighted prediction (WP), direct prediction, CABAC, and interlaced video capabilities [1, 2]. The Extended profile was developed for streaming video applications. Finally, the High profile, which was defined in 2005 by the FRExt (Fidelity Range Extension) [2] extension, provides support to different color sub-sampling (4:2:2 and 4:4:4), besides all Main profile features. The standard also defines sixteen operation levels [1, 2], which are classified in accordance to the desired processing rate. This work presents an embedded solution for motion compensation of an H.264/AVC decoder considering the Baseline profile at Level 3.

3 Motion Compensation in H.264/AVC

The operation of motion compensation in a video decoder can be regarded as a copy of predicted macroblocks from the reference frames. The predicted macroblock is added to the residual macroblock (generated by inverse transforms and quantization) to reconstruct the macroblock in the current frame.

The motion compensator is the most demanding component of the decoder, consuming more than half of its computation time [3]. Intending to increase the coding efficiency, the H.264/AVC standard adopted a number of relatively new technical developments. Most of these new developments rely on the motion prediction process, like: variable block-size, multiple reference frames, motion vector over picture boundaries, motion vector prediction, and quarter-sample accuracy. This paper will explain in more details just the features that are used in the Baseline profile.

Quarter-sample accuracy: Usually, the motion of blocks does not match exactly in the integer positions of the sample grid. So, to find good matches, fractional position accuracy is used. The H.264/AVC standard defines half-pixel and quarter-pixel

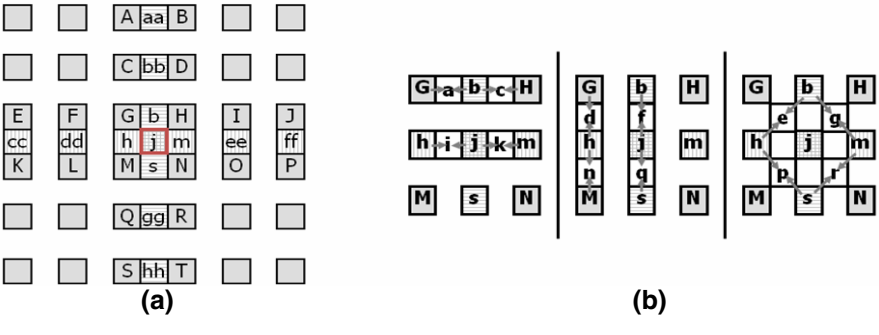

Fig. 2. (a) Half-sample luma interpolation and (b) Quarter-sample luma interpolation

accuracy for luma samples. When the best match is an integer position, just a 4x4 samples reference is needed to predict the current partition. However, if the best match is a fractional position, an interpolation is used to predict the current block. A matrix with 4x9 samples is needed to allow the interpolation of a fractionary vector in the 'X' direction, while a matrix with 9x4 samples is needed to allow the interpolation of a fractionary vector in the 'Y' direction. When the fractionary vectors occur in both directions, the interpolation needs a matrix with 9x9 samples. This need of extra samples to allow the interpolation has a direct impact on the number of memory accesses.

Fig. 2(a) shows the half-samples interpolation, which is made by a six-tap FIR filter. Then, a simple average from integer and half-sample positions is used to generate the quarter-sample positions, as shown in Fig. 2(b).

Multiple reference frames: In H.264/AVC, slices are formed by motion compensated blocks from past and future (in temporal order) frames. The past and future frames are organized in two lists of frames, called List 0 and List 1. The past and future frames are not fixed just to the immediate frames, as in early standards. Fig. 3 presents an example of this feature.

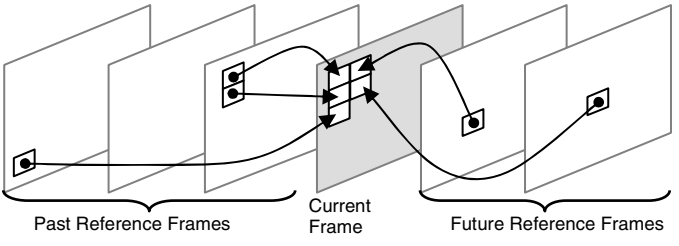

Fig. 3. Multiple Reference Frames

4 MC Hardware Accelerator Architecture

The hardware accelerator module for MC targets the bi-dimensional FIR filter, which is used in the luminance quarter-pixel interpolation process. This filter was designed using the 1-D separability property of 2-D FIR filters. Other MC filter implementations presented in the literature [6, 7, 8] use a combination of different vertical and horizontal FIR filters serially and target an ASIC implementation. In the architecture of this work, the 2-D interpolation is done by only four FIR filters used for vertical and horizontal filtering. The bilinear interpolation used to generate quarter-sample accuracy is done by bilinear filters embedded in the FIR filters.

The hardware accelerator was designed to process 4x4 samples blocks. A six-tap filter is used to generate a block of 4x4 interpolated samples. An input block of up to 9x9 samples is necessary to generate the interpolation.

The motion compensation luminance filtering considers eight different cases in this accelerator architecture, as listed below and presented in Fig. 4:

(a) No interpolation: The samples by-pass the filters;
(b) No vertical interpolation without ¼ samples: the samples pass the FIR filters once with FIR interpolation;
(c) No vertical interpolation with ¼ samples: the samples pass the filters once with FIR and bilinear interpolation;
(d) No horizontal interpolation without ¼ samples: the samples by-pass the filters and are stored in the transposition memory, then the memory columns are sent to the FIR filters once with FIR interpolation;
(e) No horizontal interpolation with ¼ samples: the samples by-pass the filters and are stored in the transposition memory, then the memory columns are sent to the FIR filters once with FIR and bilinear interpolation;
(f) Horizontal and vertical interpolations without ¼ samples: the samples pass the filters twice with FIR interpolation;
(g) Horizontal and vertical interpolations with ¼ samples: the samples pass the filters twice with FIR interpolation in the first time and with FIR and bilinear interpolation in the second one;
(h) Horizontal and two vertical interpolations with ¼ samples: the samples pass the filters three times with FIR interpolation in the first and second times and with FIR and bilinear interpolation in the third time;

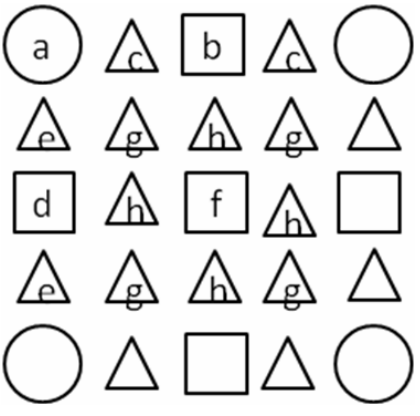

Fig. 4. Filtering cases

Fig. 5 presents the proposed MC hardware accelerator organization as well as its main modules and connections. The number above each connection indicates its width considering the number of 8-bit samples.

The first procedure to start each block processing is to set up the filtering parameters. In this case, the parameters are only the motion vector coordinates 'X' and 'Y'. This information defines the kind of filtering that will be used and spends one clock cycle. The 'X' and 'Y' coordinates are stored in two specific registers inside the architecture, which are omitted in Fig. 5.

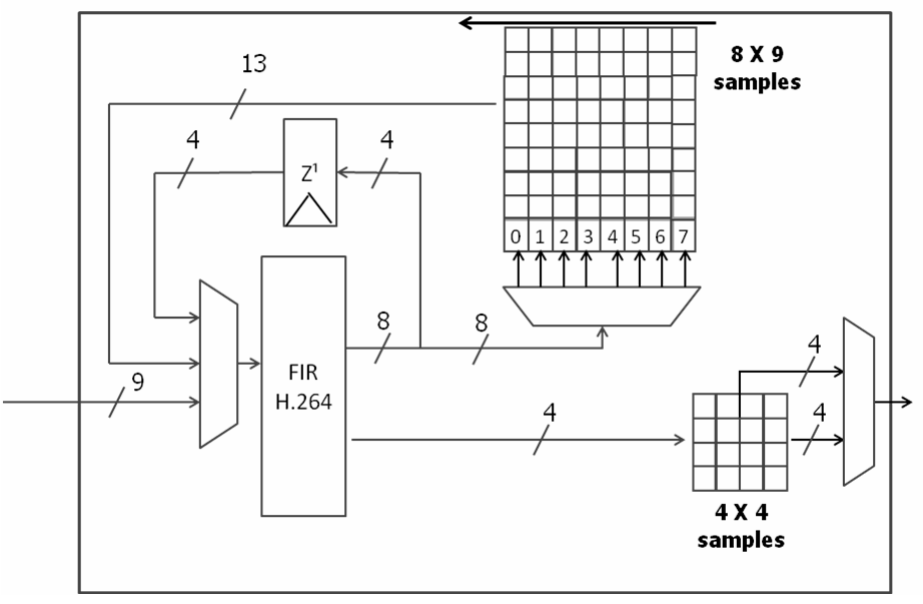

Fig. 5. MC unit architecture

The input receives a line with 9 samples, and these samples are sent to the FIR filter module. This module contains four 6-tap FIR filters working in parallel. After the interpolation, which generates four half-pixel samples, these results are sent to a transposition memory together with four non-interpolated samples. If just 1-D interpolation is needed, then the filter output is sent directly to the output memory. This process is repeated up to nine times to process all the 9x9 input block, completing the first 1-D filtering and filling the 8x9 samples transposition memory.

After filling the transposition memory by shifting the lines vertically, the columns are shifted horizontally to the left side, sending the left column samples to the filter to be interpolated in the second dimension. Each column is composed by 9 full or half-samples.

The quarter-samples are generated together with the half-samples during the same filter pass. Each filter can perform the FIR filtering and the bilinear filtering at the same clock cycle, since the bilinear filter is embedded in the FIR filter as shown in Fig. 6. However, when quarter-sample accuracy is needed, other four samples must be sent to the filters.

Depending on the filtering case, the transposition memory is filled using a different order to simplify the multiplexing logic for the FIR input. When just the half-samples need to be interpolated again in the second filter loop, they are sent to the four left memory columns (columns 0 to 3) and the full samples are sent to the four right column (column s 4 to 7). When just full samples need to be filtered in the second filter loop, these samples are sent to the left columns (columns 0 to 3) and half-samples to the right columns (columns 4 to 7). Finally, when both half and full samples need to be filtered, the columns are interleaved, even columns are filled with full samples while

odd columns are filled with half-samples (columns 0, 2, 4, and 6 for full-samples and 1, 3, 5, and 7 for half-samples).

When just half or full samples are interpolated, after the second filtering loop the results are sent to the output memory, which stores the output block of 4x4 interpolated samples. If both half and full samples must be filtered again, the full samples are processed and the outputs are stored in four delay registers to create one cycle of delay. So, in the next cycle, when the half-sample columns filtered by the FIR filter, the interpolated samples processed in the past cycle are sent to the embedded bilinear filter to generate the quarter-sample. After the interpolation is completed, the output is also sent to the output memory.

The output memory can be read by column or by lines, depending whether the input was transposed or not. The kind of output depends on the type of interpolation and is controlled through an output multiplexer.

Each FIR filter is composed by six taps with coefficients (1, -5, 20, 20, -5, 1). Fig. 6 shows the FIR filter hardware, which was designed using only additions and shifts to eliminate the multiplications. With six 8-bit inputs (E, F, G, H, I, J), the FIR block includes five adders, two shifters, one subtractor, and a clipping unit to keep the values in the range [0..255]. A bilinear filter was embedded in the FIR filter. As inputs, the bilinear filter uses the FIR output and an 8-bit input (Y) to perform the bilinear filtering.

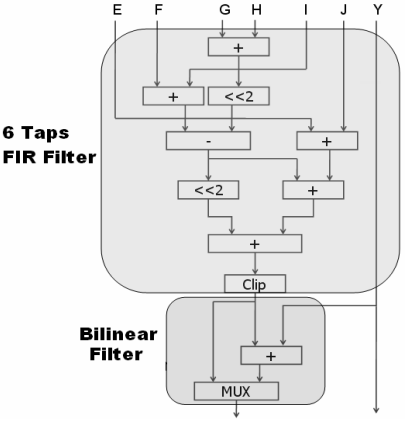

Fig. 6. FIR filter block diagram

The MC unit was described in VHDL and validated by simulation using the Mentor Graphics ModelSim software. The simulation was controlled by a testbench also written in VHDL.

5 Integration

To evaluate the designed hardware accelerator, it was integrated with a general-purpose processor. The MIPS Plasma core was chosen because of its simple RISC

organization and because its VHDL description was available in the "Opencores" community [9]. The MIPS Plasma organization and its integration with the MC hardware accelerator module can be seen in Fig. 7.

The MIPS Plasma processor is fully compliant with the MIPS I(TM) ISA, but without support to non-aligned memory access instructions. It was designed using a Von Neumann architecture with 32x32-bits registers and three pipeline stages.

The integration demanded some modifications on the MC accelerator and on the MIPS Plasma core. Once the processor data bus is 32-bits wide and the input of MC is 72-bits wide, a hardware wrapper was designed to make them compatible. Finally, some changes in the processor control block were made to insert new instructions and to call the new module tasks.

MC spends a variable number of cycles to process a 4x4 block, and the Plasma processor does not support parallel instructions execution. Therefore, the processor pipeline is kept frozen while this module is working.

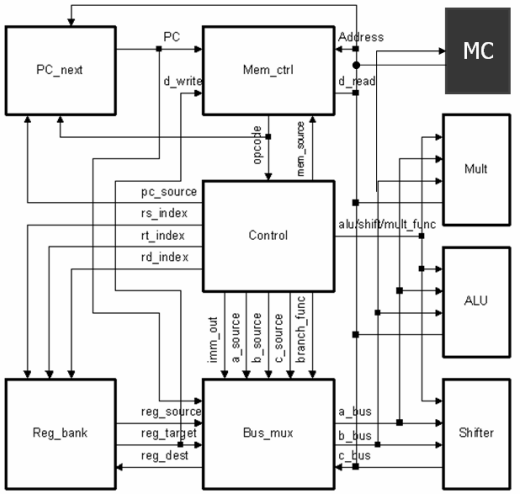

Fig. 7. Integration architecture among MIPS Plasma and MC unit

The MC unit architecture uses an input interface modeled as a ping-pong buffer (see Fig. 8), which receives a 32-bit word per cycle, storing up to three words. After the module received the appropriate number of words, a specific instruction sends the signal to start the processing.

Each word is read from the Plasma memory, sent to the processor register bank, and finally sent to the ping-pong buffer. This happens up to three times to process each MC input line. Finally, the words are sent to the MC accelerator. The ping-pong buffer filling process happens up to nine times for each block. Many clock cycles are spent to feed the MC unit. After loading, the data processing occurs relatively fast. After the processing, the results can be read from MC registers to the Plasma register bank.

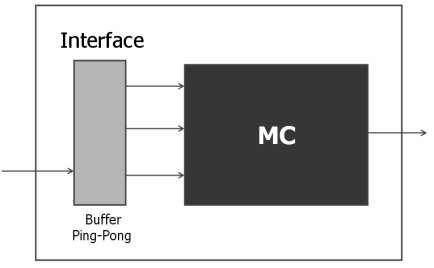

Fig. 8. Ping-Pong buffer

Some new instructions were added to the MIPS Plasma instruction set to control the MC module. Each operation of reading, writing, or setting the MC hardware accelerator originated a new processor instruction. The new instructions use a MIPS type R format (as shown in Fig. 9) composed by a 6-bit op-code field, two 5-bit source register index fields, one 5-bit target register index field, and one 6-bit function field.

The new instructions use in the op-code field the value "111111", while in the function field the values from "000000" to "0000100" were used. This op-code value is reserved for eventual instruction set expansions. The new instructions are listed in Table 1. The MC_WRITE instruction uses the "Rt" field to indicate the source register, while the MC_READ instruction uses the "Rd" field to point the target register. The other register fields need no specific value.

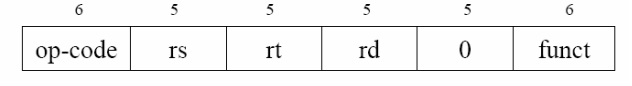

Fig. 9. Type R MIPS Instruction

Table 1. New Instructions

Function	Name	Description
000000	MC_SET	Sets motion vector coordinates
000001	MC_WRITE	Writes a word
000010	MC_PROC	Starts the filtering
000100	MC_READ	Reads a word

The final integration step was the validation of the integrated modules. An assembly code was written using the new instructions to feed and control the MC hardware accelerator. This assembly was loaded to the Plasma ROM memory, and its correct execution was verified through simulation using the Mentor ModelSim software.

6 Results and Comparisons

The MIPS Plasma processor and the MC accelerator architectures were synthesized targeting a Xilinx Virtex-2 PRO FPGA device (XC2VP30-7) [10] using the Xilinx ISE 8.1 software.

Table 2 shows the resource utilization for the Plasma processor and the MC module in the second and third columns, respectively. The fourth column presents the synthesis results for the processor integrated to the MC hardware accelerator. Finally, the last column shows the ratio between synthesis results obtained for the Plasma version with the hardware accelerator and for its original version. The synthesis results presented an important increasing in hardware resource utilization besides a degradation in the maximum operation frequency. The high increasing in register utilization occurs because the memories inside the MC module were implemented as banks of registers and not as Block-RAMs available in this FPGA family.

Table 2. Synthesis Results

	Plasma	**MC**	**MC + Plasma**	**Increase**
LUTs	2599	1594	3966	52 %
Reg	402	758	1262	213 %
Slices	1378	891	2172	57 %
Freq.	~90MHz	~74 MHz	~74 MHz	-21%

Two different software codes were described to compare the performances of the standard MIPS Plasma and the modified MIPS Plasma. The first code makes the motion compensation task in software without any hardware acceleration. The second solution is a HW/SW solution using the hardware acceleration instructions to call the new MC module. For a fair comparison, the software solution was optimized to have no multiplications, since a multiplication spends 32 clock cycles in this processor. Additions and shifts were used to eliminate multiplications.

The first software solution (without MC accelerator) was described in C language based on the H.264/AVC reference software (JM 12.2) [11]. GCC was used as a cross compiler to generate MIPS machine code. This machine code was mapped to the MIPS Plasma ROM memory. The software was analyzed through simulations using Mentor Graphics ModelSim 6.0. These simulations were used to count the number of clock cycles necessary to process a 4x4 samples block at each different interpolation case.

The HW/SW solution demanded an assembly description to use the new instructions. The same method of simulation used in the first code was applied to the HW/SW solution. Another way to generate the code using MC accelerating instructions is adapting the GCC compiler to use these instructions, but this solution was not implemented in this paper.

The results obtained in the simulation process are shown in Tables 3 and 4. These tables present a considerable increase in performance with the use of the MC acceleration hardware. The performance increase reaches more than 95% in clock cycles and 94% in execution time, when comparing the average gains of the HW/SW solution in

relation to the SW one. As expected, because of the simplicity of the Plasma processor, the increase in area was relatively high and the performance gains were expressive.

The first and second columns of Tables 3 and 4 present the different interpolation cases and their probability of occurrence.

In Table 3, the third column presents the total number of clock cycles spent to process each kind of block using a SW solution. The three following columns show the number of cycles spent to process a block in the HW/SW solution, considering also the cycles used for memory accesses and for effective processing. The seventh column presents the percentage of reduction in number of clock cycles when using the MC accelerator.

Table 4 shows the total execution time for the SW and HW/SW solutions in the third and fourth columns, respectively. The last column presents the percentage of reduction in terms of execution time, considering 90 MHz for the SW solution and 74 MHz for the HW/SW one.

Table 3. Results and Comparison (clock cycles)

Interpolation Cases	Prob.	SW	HW/SW			Clock Cycles Reduction
		Total # of Cycles	Memory Cycles	Processor Cycles	Total # of Cycles	
No Interpolation	1/16	187	24	8	32	82.89%
No Vertical S/1/4	1/16	802	44	8	52	93.52%
No Vertical C/1/4	1/8	1069	44	8	52	95.14%
No Horizontal S/1/4	1/16	811	62	17	79	90.26%
No Horizontal C/1/4	1/8	1084	62	17	79	92.71%
Vert. & Hor. S/ 1/4	1/16	2245	62	17	79	96.48%
Vert. & Hor. C/ 1/4	1/4	1717	62	17	79	95.40%
Vert. & 2 Hor. C/ 1/4	1/4	2667	62	21	83	96.89%
Weighted Average	-	**1617.9**	**56.25**	**15.75**	**72**	**95.55%**

Table 4. Results and Comparison (execution time)

Interpolation Cases	Prob.	SW	HW/SW	Total Time Reduction
		Time (ns)	Time (ns)	
No Interpolation	1/16	207.78	43.24	79.19%
No Vertical S/1/4	1/16	891.11	70.27	92.11%
No Vertical C/1/4	1/8	1187.78	70.27	94.08%
No Horizontal S/1/4	1/16	901.11	106.76	88.15%
No Horizontal C/1/4	1/8	1204.44	106.76	91.14%
Vert. & Hor. S/ 1/4	1/16	2494.44	106.76	95.72%
Vert. & Hor. C/ 1/4	1/4	1907.78	106.76	94.40%
Vert. & 2 Hor. C/ 1/4	1/4	2963.33	112.16	96.22%
Weighted Average	-	**1797.71**	**97.30**	**94.59%**

7 Conclusions

This work presented a new architectural solution for a hardware accelerator for the motion compensation of an H.264/AVC Baseline Profile video decoder. The applicability in embedded devices was demonstrated. The MC accelerator was validated and successfully integrated to the MIPS Plasma VHDL description. Through simulations, data were extracted to evaluate the performance increase of the proposed solution. Results indicate sufficient performance to execute the luminance motion compensation decoding task in real time for H.264/AVC Baseline Profile at level 3. H.264/AVC at level 3 demands decoding 525SD (720x480) video sequences at 30 fps or 625SD (720x576) video sequences at 25 fps. The HW/SW performance gains were compared to a SW solution running in the MIPS Plasma processor. These results indicate a reduction of 95% in the number of necessary clock cycles and a reduction of 94% in execution time when using the MC accelerator.

This architecture working at 74MHz and using an average number of 72 clock cycles to decode each 4x4 block can process up to 64.2K P-type macroblocks (16x16 samples) per second, reaching an average processing rate of 39.6 P-frames per second for 625SD (720x576).

References

1. JVT, Wiegand, T., Sullivan, G., Luthra, A.: Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec.H.264 ISO/IEC 14496-10 AVC). JVT-G050r1, Geneva (2003)
2. International Telecommunication Union.: Advanced Video Coding for Generic Audiovisual Services. ITU-T Recommendation H(264) (2005)
3. Wiegand, T., Schwarz, H., Joch, A., Kossentini, F., Sullivan, G.: Rate-constrained Coder Control and Comparison of Video Coding Standards. IEEE Transactions on Circuits and Systems for Video Technology 13, 688–703 (2003)
4. International Telecommunication Union: Generic Coding of Moving Pictures and Associated Audio Information - Part 2. ITU-T Recommendation H(262) (1994)
5. International Organization For Standardization. Coding of Audio Visual Objects - Part 2 ISO/IEC 14496-2 - MPEG-4 Part 2 (1999)
6. Azevedo, A., Zatt, B., Agostini, L., Bampi, B.: Motion Compensation Decoder Architecture for H.264/AVC Main Profile Targeting HDTV. In: IFIP International Conference on Very Large Scale Integration, VLSI SoC, Nice, France, pp. 52–57 (2006)
7. Wang, S.-Z., Lin, T.-A., Liu, T.-M., Lee, C.-Y.: A New Motion Compensation Design for H.264/AVC Decoder. In: International Symposium on Circuits and Systems. In: ISCAS, Kobe, Japan, pp. 4558–4561 (2005)
8. Chen, J.-W., Lin, C.-C., Guo, J.-I., Wang, J.-S.: Low Complexity Architecture Design of H.264 Predictive Pixel Compensator for HDTV Applications. In: Proc. 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP, Toulouse, France, pp. 932–935 (2006)
9. OPENCORES.ORG (2007), Available from: URL: http://www.opencores.org/projects.cgi/web/ mips/overview
10. Xilinx Inc. (2007), Availabe from: http://www.xilinx.com
11. H.264/AVC JM Reference Software (2007), Available from: URL: http://iphome.hhi.de/suehring/tml