Real-Time Architecture for HEVC Motion Compensation Sample Interpolator for UHD Videos

Wagner Penny, Guilherme Paim, Marcelo Porto, Luciano Agostini, Bruno Zatt
Federal University of Pelotas
Group of Architectures and Integrated Circuits
Pelotas, Brazil
{wi.penny, gppaim, porto, agostini, zatt}@inf.ufpel.edu.br

ABSTRACT
This paper presents a high throughput architecture for a Motion Compensation (MC) sample interpolator targeting the High Efficiency Video Coding (HEVC) standard. Real-time operation and low power dissipation in video coding systems have become important research challenges, especially in mobile devices with limited computational resources and battery availability. The Fractional Motion Estimation (FME) is one of the tools in the inter-frame prediction module, which has the goal of reducing temporal redundancies by capturing motion more accurately. The Motion Compensation, responsible for compensating the motion detected at the FME process, demands the major computational effort at the decoder side and represents one of the main challenges when developing real-time HEVC decoders. The proposed architecture is able to save hardware resources through an optimized filter organization and is capable to decode UHD 4320p@60fps video sequences in real time when synthesized for the TSMC 65nm standard-cell library.

Categories and Subject Descriptors
B.5.1 [**Register-Transfer-Level Implementation**]: Design - *Datapath Design*.

B.7.1 [**Integrated Circuits**]: Type and Design Styles - *Algorithms Implemented in Hardware.*

General Terms
Algorithms, Performance, Design.

Keywords
Motion compensation, sample interpolator, video coding, HEVC decoder, hardware architectures.

1. INTRODUCTION
The latest research and standardization efforts in video coding have led to a new specification, called High Efficiency Video Coding (HEVC) [1]. HEVC is the new state of art in video coding and reaches 50% bit rate reduction compared to its predecessor, H.264 [2], keeping the same objective quality whereas improving subjective quality [3]. However this new standard significantly increases computational complexity compared to H.264 [4].

In both standards, fractional motion estimation (FME) is used in the inter-prediction blocks to reduce temporal redundancies by more accurately capturing motion [5]. Basically, FME applies an interpolation over the integer samples in the reference block in order to generate sub-pixels (fractional) samples and allow finer grain motion estimation.

One of the blocks within the FME is the Motion Compensation (MC) used to obtain the reconstructed samples. In the MC, during the decoding process, the interpolation block is responsible for the major complexity of the MC. HEVC defines 7- or 8-tap FIR (finite impulse response) filters for calculating fractional samples, depending on their positions and 4-tap FIR filters for fractional chrominance samples.

Due the great complexity of the MC, hardware solutions are strongly desirable to ensure high throughput and energy efficiency. There are several works in the literature which aim the development of hardware solutions. Pastuszak [1] proposes an architecture for interpolation targeting HEVC encoder which gives high throughput. The design can be used instead of the interpolator described as input and output interfaces. That work proposes two versions of the architecture to show its resource consumption and performance. The both implementations apply different resources for filter multiplications, using dedicated units or replacing them by adders and subtractors. Goto [2] also proposes an architecture for HEVC interpolation which reaches good results. The proposed architecture uses a reconfigurable filter to reduce the implementation area of MC interpolation and an optimized pipeline organization for the interpolation engine. That work also proposes a filter reuse scheme, allowing the improvement of the parallelism. Zatt [6] proposes a high throughput architecture for H.264/AVC motion compensation sample interpolator, which uses an efficient pipeline structure. Note H.264 uses 6-tap FIR for luminance half-samples and bilinear filters for quarter-samples and chrominance samples. Although the solutions found in the literature are promising, some limitations are detected. The solution [1] demands excessive hardware and cannot achieve high resolutions. The work [2] uses 8-tap FIR filters for luminance samples, which are not compliant with HEVC - 7-8 taps are used in the HEVC. The architecture proposed in [6] achieves a high throughput, but only for the predecessor standard H.264 and is unable to reach Ultra High Definition (UHD) resolutions.

This paper presents a high throughput architecture for the HEVC MC interpolation using pipeline and hardware reuse. The proposed architecture uses HEVC-compliant filters and reduces memory communication in comparison with related works by adopting larger

SBCCI '15, August 31 - September 04, 2015, Salvador, Brazil © 2015 ACM. ISBN 978-1-4503-3763-2/15/08…$15.00
DOI: http://dx.doi.org/10.1145/2800986.2801024.

block sizes as basic processing units. This architecture was described in VHDL and synthesized for TSMC 65nm standard-cell library and mapped to an Altera FPGA. This architecture can reach real time for UHD videos with a low hardware cost.

The organization of this paper is as follows. The coding structure in HEVC is briefly explained in Section 2. In Section 3 it is discussed the selection of a basic processing unit. Section 4 shows the proposed interpolator architecture whereas Section 5 presents the obtained results. Finally, Section 6 presents the conclusions.

2. CODING STRUCTURE IN HEVC

In HEVC the video compression structure defines three block concepts: Coding Unit (CU), Prediction Unit (PU) and Transform Unit (TU), which allow many optimization techniques. The image is divided in Coding Tree Units (CTU), which contains the luminance and chrominance blocks. The CTU can be divided in CUs and these in PUs. The prediction units have information about prediction modes, motion vectors, and reference blocks indices.

Each CTU contains a quad tree syntax, also referred as coding tree, which specifies its subdivision into CUs. A CU consists of a square block of luma samples, the two corresponding blocks of chroma samples and the syntax associated with these samples blocks. The luma and chroma sample arrays that are contained in a CU are referred to as coding blocks (CB). Using 4:2:0 chroma sampling format, each 2Nx2N luma CB is associated with two 2N/2 x 2N/2 chroma CBs. The minimum size of CUs is signaled in the sequence parameter set, it can range from 8x8 luma samples to the size of CTU (maximum size 64x64). The PUs size can range from the minimum size (4x8 or 8x4) to the maximum CU size (64x64), allowing asymmetric partitioning, taking a rectangular shape [5].

The motion compensator is the most demanding component of the decoder, consuming more than half of its computation time [6]. This module is composed by three main structures: the Motion Vector (MV) predictor, which infers the motion vectors using syntax elements available in the bitstream and information from neighboring blocks; the sample processor, which generates half or quarter samples; and the frame memory access, which brings the reference frames data to be interpolated by sample processing structure.

The focus of this work is the sample processor. With the information about the motion vector and reference samples, the interpolator is able to calculate the fractional samples, however, in motion compensation, only the necessary samples need to be found, which contrasts with the motion estimation step, whereas it evaluates all fractional samples. Figure 1 shows an example of a motion vector, which points to (a) an integer position and to (b) a fractional position.

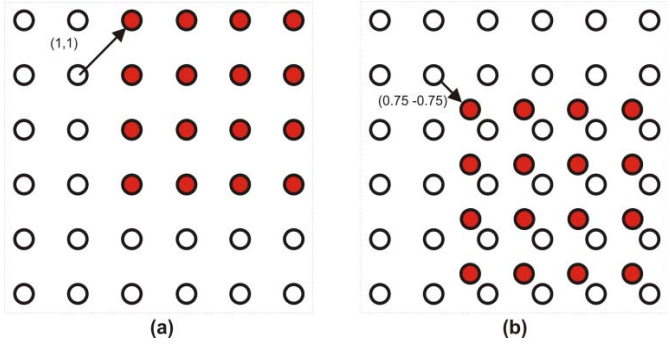

Figure 1. (a) MV-Integer precision (b) MV-Quarter precision.

HEVC kept the same level of precision from H.264 and did the interpolation process more accurately and allowed higher level of parallelism. Figure 2 presents all the fifteen fractional samples that can be calculated.

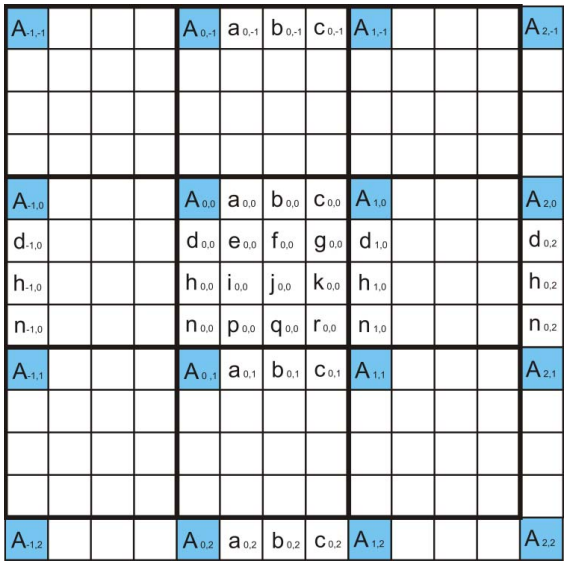

Figure 2. Fractional luminance samples.

Depending on the motion vector information, only some samples need to be calculated. Each sample can be evaluated using a FIR filter with 7 or 8 tap. The filters classification is shown on Figure 3 and depends of the sample position.

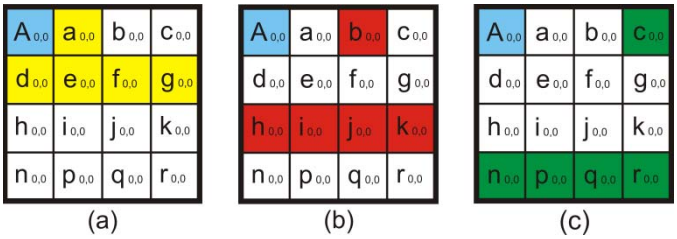

Figure 3. (a) Up filter, (b) Middle filter and (c) Down Filter.

Basically, there are two situations, a horizontal filtering or a vertical filtering. In the first one, the samples *a*, *b* and *c* are evaluated using as input the reference integer samples. In the second ones, all the others samples are evaluated, the only difference is that samples *d*, *h* and *n* are calculated using the reference integer samples and the other ones using samples which were calculated by the horizontal filtering. Samples *e*, *i* and *p* use *a* samples, *f*, *j* and *q* use *b* samples and *g*, *k* and *r* use *c* samples. In Table 1 are presented the filter coefficients according to each type.

Table 1. Filter Coefficients for Luma Interpolation

Filter	Coefficients
UP	{-1, 4, -10, 58, 17, -5, 1}/64
MIDDLE	{-1, 4, -11, 40, 40, -11, 4, -1}/64
DOWN	{1, -5, 17, 58, -10, 4, -1}/64

The chrominance filter is a 4-tap FIR filter, which uses the coefficients present in Table 2, according to the desired precision, given by the motion vector.

Table 2. Filter Coefficients for Chroma Interpolation

Filter Precision	Coefficients
1/8	{-2, 58, 10, -2}/64
1/4	{-4, 54, 16, -2}/64
3/8	{-6, 46, 28, -4}/64
1/2	{-4. 36, 36, -4}/64

3. BASIC PROCESSING UNIT: A CASE STUDY

The motion compensation must support all possible PU sizes defined in HEVC. This requirement represents an important challenge when considering memory access pattern, control and processing parallelism. Thus, to deal with this challenge it is possible to decompose all PU sizes in basic processing units (smaller blocks) leading to a more regular processing and memory access. Related works have used this approach decomposing all blocks into the smallest one. In [6], that focuses H.264, the smallest possible blocks size - the 4x4 - was used as basic processing unit.

The smallest PU sizes defined in HEVC are 4x8 and 8x4 and these blocks are good candidates for basic processing units. However, these rectangular blocks present distinct number of samples per line. As our architecture is planned to process one line per cycle (see Section 4), rectangular blocks would lead to irregular behavior. Consider the case where eight samples are processed per cycle, when 4x8 blocks are consumed half of the operator are unused. Additionally, the memory interface would provide four useless samples. Therefore, to better exploit the operators and keep control and memory regularity, square-shaped blocks are desirable.

In this work we evaluate the impact of selecting three sizes as basic processing unit: 4x4, 8x8, and 16x16. Three parameters are considered to select the best basic unit size: (i) Memory communication – amount of data required from memory; (ii) Samples to process – amount of samples to be interpolated including the overhead of samples processed but not required; (iii) Line-level parallelism.

To properly understand the impact of the basic unit size on the memory access it is necessary to understand how the interpolation filters behave. Figure 4 presents the data necessary to interpolate an 8x8 block. The black boxes represent the 8x8 central samples used for interpolation whereas gray boxes represent the borders necessary to apply the 8-tap filter. Generically, filters with *n*-taps, where *n* is an even number, require *n-1* sample borders.

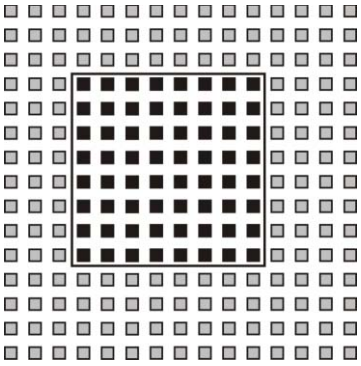

Figure 4. Reference luminance samples.

Observe in Figure 5 the 7-samples borders for both, horizontal and vertical directions. These borders have always the same size, independently from the block size to be processed. It means that 8x8 block size requires a reference sample matrix of 15x15 (Figure 4). So, this block requires 225 reference samples to predict 64 samples. If 4x4 blocks are used to interpolate the same 64 samples, four 11x11 reference matrixes would be required demanding 484 reference samples. This represent the memory retransmission of overlapping borders as depicted in Figure 5. Observe the four neighboring blocks colored as red, yellow, blue, and green, respectively. There is a large region of overlapped references, as depicted by the shaded areas.

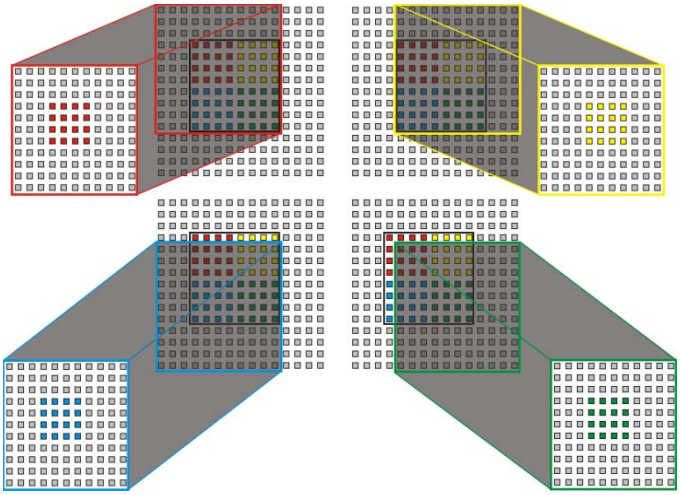

Figure 5. Overlapping edges among 4x4 neighboring blocks.

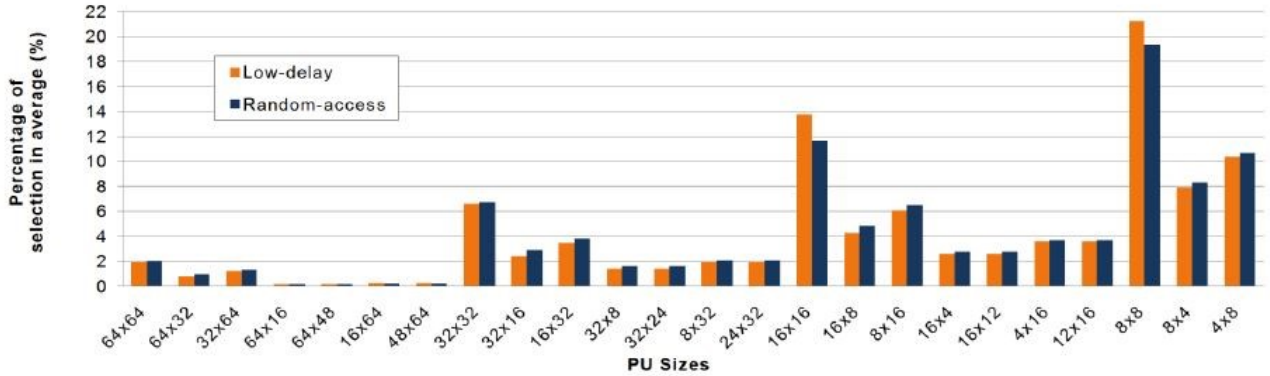

Figure 6. Percentage of selection of PU sizes in average [8]

At the perspective of data to be processed, when the basic unit is larger than the PU to be processed part of the filtered samples are useless. Consider a basic unit of size 8x8 and a PU of 8x4. In this case, 64 samples would be filtered (8x8) but half of the samples are discarded generating processing overhead. This phenomenon does not happen when the basic unit is equal or smaller than the PU. Observe that the processing overhead is proportional to the area covered by PUs smaller than the basic processing unit. The larger the basic unit, the larger the processing overhead.

Table 3 presents a case study considering three basic unit sizes considering one 1080p frame at 30fps. To estimate the memory communication and samples to be processed the PUs representativeness presented in [8] is used (see Figure 6). It is clear to notice that as the basic unit grows larger the memory communication is reduced as less borders need to be retransmitted. At the other hand, the increase in basic unit size leads to higher processing overhead. This happens because there are more PUs smaller than the basic units incurring in more discarded samples. In a closer discussion, the use of 4x4 basic blocks requires no reprocessing, 8x8 basic blocks discard 50% of data when processing 8x4 and 4x8 PUs (it means that 64 samples are processed to deliver 32 samples of 8x4 or 4x8 blocks). Finally, if 16x16 basic units are used 87.5% of data is discarded for 8x4/4x8 PUs, 75% for 8x8 PUs and 50% for 16x8/8x16 PUs. It is worth to mention that similar behavior is observed in chrominance channel.

Considering the presented case study, we have selected the 8x8 basic processing unit for the hardware architecture described in Section 3. As 4:2:0 color subsampling is used, the chrominance channels use 4x4 basic processing units.

Table 3. Basic Processing Unit Definition

Average for one 1080p frame	Basic Processing Unit		
	4x4	8x8	16x16
Memory Communication (ksamples/s)	81.393	39.648	46.569
Samples to Process (Msamples/s)	62.21	63.24	83.84
Line-level Parallelism	4	8	16

4. INTERPOLATOR ARCHITECTURE

The designed architecture for luma interpolation uses 8 horizontal filters, an intermediary pipeline before the vertical filtering and 8 vertical filters. Figure 7 shows the luminance interpolator architecture. The new architecture explores the pipelining and hardware reuse, leading to an optimized parallelism and performance. Each sample is represented as a black square on the figure, they feed horizontal (H) filters, which is the first step of interpolation filters. Next, the samples feed vertical (V) filters. H and V blocks are detailed in the Figure 7.

Inside a horizontal filter there are two filters: an up/down and a middle filter. The same structure is found inside a vertical filter. Depending on motion vector information a mux defines the data path.

The chrominance interpolation architecture uses 4 horizontal block filters, an intermediary pipeline before the vertical filtering and 4 vertical filters. These structures work in a similar way compared to luma interpolator. Figure 8 shows the chrominance interpolator architecture.

At each cycle 15 luma reference samples are read, corresponding to each line of the reference matrix, these samples are the inputs of 8 luma horizontal filters, which have 8 input signals each. These filter blocks have, each one, a bypass, if motion vector in x axis (MVX) points to an integer position, an up/down and a middle filter, if MVX points to a fractional position (a mux select the correct output based on MVX information). The luma horizontal filter output is connected to a pipeline with 8 stages. On each cycle each position of the pipeline is forwarded, after eight cycles the pipeline is filled, so all the positions of the pipeline feed the other block filters, called luma vertical filters. These block filters have, each one, a bypass, if motion vector in y axis (MVY) points to an integer position, an up/down and a middle filter, if MVY points to a fractional position (a mux select the correct output basing on MVY information).

The output samples pass through a temporal barrier before going out. The outputs may have integer samples, half or quarter factionary samples. In the worst case, the system latency is 8 cycles (from the intermediary pipeline). After that, supposing a continuous input, every 15 cycles deliver 64 samples. So, the basic processing unit, the 8x8 block, can be built, which gives a throughput of 4.26 samples per cycle.

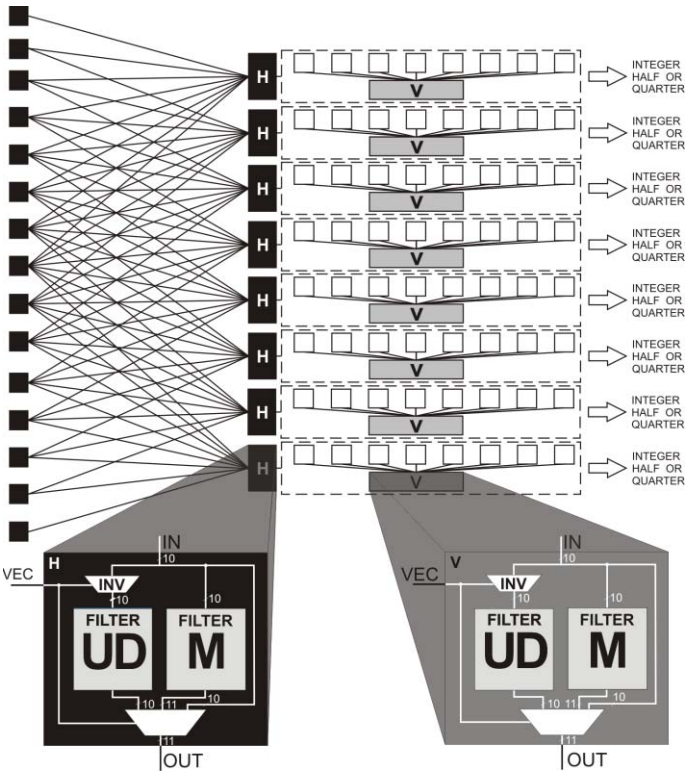

Figure 7. Luminance interpolator designed architecture.

On each cycle are read 7 chroma Cb reference samples and 7 chroma Cr reference samples, corresponding to the first line of the reference matrices, these samples are the inputs of 4 block filters, called chroma horizontal filters, which has 4 input signals. These block filters have, each one, a bypass and 4 filter structures, depending of MVX information one of them is selected. The chroma horizontal filter output is connected in a pipeline with 4 stages. On each cycle, each position of the pipeline is forwarded when the pipeline is filled,

so all the positions of the pipeline feed the other block filters, called chroma vertical filters. These block filters have, each one, a bypass and 4 filter structures, depending of MVY information one of them is selected. For luma interpolation only the two least significant bits from MV are important (quarter precision) and for chroma interpolation the three least significant bits are important (eight precision).

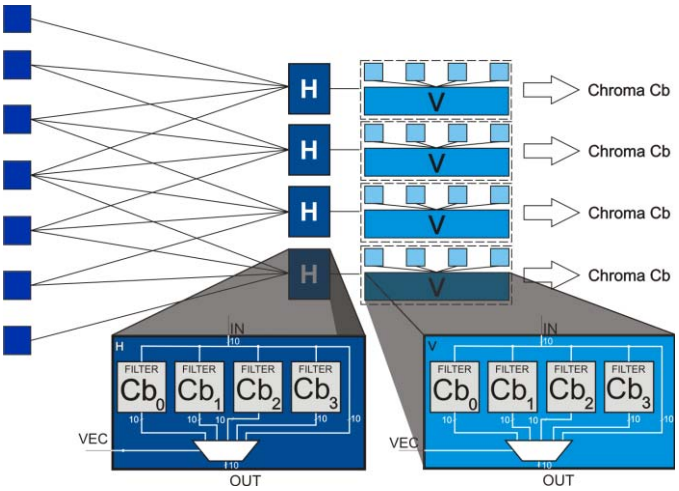

Figure 8. Chrominance interpolator designed architecture.

4.1 Filters Design

According to [7], the filters architecture can be optimized by replacing the multipliers with adders and shifters. The filters up and down use the same hardware, needing only the change of the sequence from input samples. Thereby, there are two hardware structures: an up/down filter and a middle filter. Figure 9 shows the up/down filter architecture.

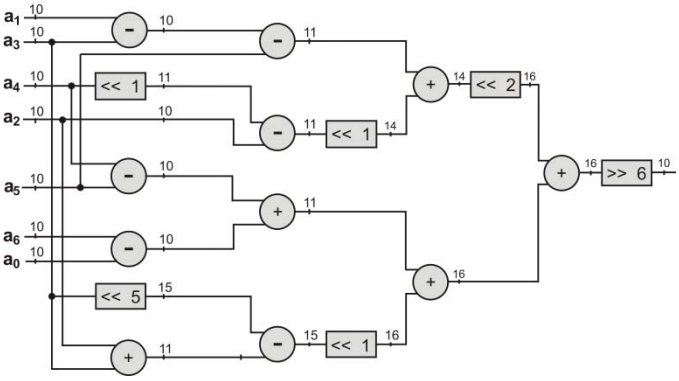

Figure 9. Up/Down filter architecture.

The up/down is a 7-tap FIR filter, which uses the same hardware, the only difference between them is the sequence of the coefficients, if filter up starts from left to right, the down filter starts from right to left. The same structure can be described in VHDL, changing the port map coefficients sequence of each structure, depending if then will be used as an up or down filter. The middle is an 8-tap FIR filter and its structure can be seen on Figure 10.

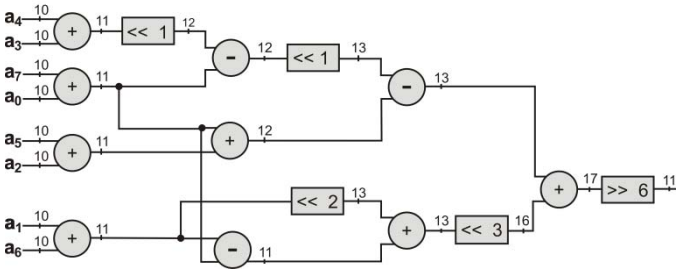

Figure 10. Middle filter architecture.

5. SYNTHESIS RESULTS

The proposed architecture was described in VHDL and synthesized using a 65nm TSMC standard cell library with the Synopsys Design Compiler tool. Our work was also mapped to an Altera Stratix V GX 5SGSED6K1F40C2 FPGA device.

The ModelSim 10.1 tool was used to simulate and to validate the architecture design. The architecture was validated through codes evaluation in Matlab. Table 4 presents the synthesis results for FPGA. The frequencies obtained allow operation even at UHD 4320p (8K) resolution with 60 fps. In smaller resolutions frequency scaling could be used to save energy.

Table 4. Results for FPGA technology

Max. Frequency (Mhz)		423.37
Logic Cells (ALMs)		5093
Registers		11539
Throughput (samples/cycle)		4.26
Throughput (fps)	HD 720p	966 fps
	Full HD 1080p	429 fps
	UHD 2160p (4K)	107 fps
	UHD 4320p (8K)	60 fps

The Table 5 presents the results of filters compared with three related works, one for H.264/AVC and two for HEVC. Table 6 presents the synthesis results for ASIC. The gate count is calculated based on 2-input NANDs count and the power dissipation is generated with a power supply of 0.72V in specified target frequencies.

Some characteristics of a video reproduction are essential, such as resolution and rate processing. Considering the HD 1080p resolution (1920x1080 pixels), to achieve real time, the processing of at least 30 fps is recommended. For UHD 4K resolution (3840x2160 pixels), the recommended frame rate is at least 60 fps. This increase in the frame rate is recommended to keep a good visual motion sensation, due to the improvement of viewers perceptions given by the resolution increase. Considering the achieved frequency of 495.5 MHz, the architecture is able to process 60 fps in UHD 8K resolution, and 429 fps in HD 1080p resolution. Those results show that the developed architecture supports operating it on lower frequencies. That allows using frequency scaling to save energy, keeping the necessary performance for real-time processing of high and ultra-high definition videos.

Table 5. TSMC Synthesis Results and Comparison with Related Works

Related Work	Zatt et. Al.[6]	Pastuszak et. Al[1]	Goto et. Al.[2]	Developed
Standard	H.264	HEVC	HEVC	HEVC
Technology	TSMC 180nm	TSMC 90nm	TSMC 90nm	TSMC 65nm
Max. Frequency (MHz)	129,9	400	171	495.5Mhz
Gates (K)	12.34	211.693	32.49	98.68
Max. Throughput	1080p@30fps	1080p@30fps	2160p@60fps	4320p@60fps

On Table 5 it is presented a comparison between the proposed work and other three related works. The work [6] developed an architecture for H.264 and achieves throughput only for HD 1080p resolution at 30 fps. The works [1] and [2] developed architectures for HEVC, [1] demands increased hardware resources whereas [2] achieves good results with a TSMC 90nm, however can work only at 2160p (4K) at 60 fps. Our architecture can achieve throughput to work with 4320p at 60 fps. The proposed design achieves 495.5MHz, the highest maximum frequency between the compared works.

Table 6 shows the power dissipation for frequencies targeting different resolutions at 60 fps and 0.72V. Scaling the frequency for lower resolutions saves energy. The related works do not provide information regarding power dissipation.

Table 6. Power Dissipation from TSMC process

1080p@60fps	Freq. (MHz)	30.02
	Power (mW)	2.95
4K@60fps	Freq. (MHz)	120.33
	Power (mW)	8.10
8K@60fps	Freq. (MHz)	490.19
	Power (mW)	29.12

6. CONCLUSIONS

This paper presented a hardware architecture for the sample interpolator in motion compensation. Strategies such as using basic processing unit, efficient pipelining, hardware reuse, and frequency scaling are used to increase throughput whereas reducing power dissipation.

The presented architecture is able to run at 495.5 MHz when synthesized using a 65nm TSMC standard cell library and the results are compared with related works. The proposed design has the highest processing rates among the related works supporting 4320p@60fps. This work was the only one to present results for power dissipation. The developed architecture is able to process 1080p@60fps videos with 2.95mW and 4320p@60fps with 29.12mW.

The developed architecture can achieve real time for many resolutions, including Ultra High Definition 8K, allowing frequency scaling to reduce power dissipation when lower resolutions are processed.

7. ACKNOWLEDGMENTS

We have a special acknowledgement to CNPq, CAPES and FAPERGS to support this work.

8. REFERENCES

[1] Pastuszak, G., Trochimiuk M., “Architecture Design and Efficiency Evaluation for the High Throughput Interpolation in the HEVC Encoder”. 16th Euromicro, 2013, pp.423-428.

[2] Goto et. al., “An optimized MC interpolation architecture for HEVC”. ICASSP, 2012, pp.1117-1120.

[3] Grois, D. et. al. "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders". PCS, 2013, pp. 394-397.

[4] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Performance and Computational Complexity Assessment of High Efficiency Video Encoders," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 22, pp. 1899-1909, 2012.

[5] Sullivan, G. et al. High Efficiency Video Coding (HEVC). Springer, 2014.

[6] Zatt, B. et. al. “High Throughput Architecture for H.264/AVC Motion Compensation Sample Interpolator for HDTV”. SBCCI, 2008, pp. 228-232.

[7] Afonso, Vladimir et. al. "Low Cost and High-Throughput FME Interpolation for the HEVC Emerging Video Coding Standard". IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS), 2013, Cusco. 2013 IEEE 4th Latin American Symposium on Circuits and Systems (LASCAS). p. 1..

[8] Maich, Henrique e. al. "HEVC Fractional Motion Estimation Complexity Reduction for Real Time Applications". IEEE 5th Latin American Symposium on Circuits and Systems (LASCAS), 2014.

[9] Diniz, Cláudio et. al. "A Reconfigurable Hardware Architecure for Fractional Pixel Interpolation in High Efficiency Video Coding". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCSVT), vol. 34, NO. 2, February 2015, pp. 238-251.

[10] S. Wang, D. Zhou and S. Goto, "Motion compensation architecture for 8K UHDTV HEVC decoder," IEEE International Conference on Multimedia and Expo (ICME), 2014.