International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

FPGA-based implementation of intra prediction module for HEVC decoder

Manel Kammoun, Ahmed Ben Atitallah

Abstract: ***The High Efficiency Video Coding (HEVC) is the new standard which is designed to support High Definition (HD) and Ultra-HD video cotenants. In HEVC, several new coding tools are adopted in order to improve the coding efficiency and compression ratio but with a significant increase in the computational complexity comparing to the previous standard H.264/AVC. In this paper, we focus on reducing the complexity of the most consuming block in the HEVC decoder standard which is the intra prediction module. In this context, we propose an optimized hardware architecture dedicated to support the 34 modes of intra prediction module considering 4×4, 8×8 and 16×16 block sizes. The proposed design exploits the symmetric property between horizontal and vertical modes. Hence, we implement a new hardware architecture that factorizes the same hardware resources for both directions which leads to save the hardware cost, the power consumption and the processing time. Furthermore, the different block sizes are implemented independently in order to avoid memory overhead while accessing to the shared memory. The implemented design using Xilinx Zynq-based FPGA platform can process in real time the Ultra-HD video frame of resolution (4096×2048) at 232 MHz. As well, the synthesis results using the TSMC 180 nm CMOS technology provide similar performance than our FPGA implementation. Finally, the HW/SW implementation of full HEVC decoder can process the decoding of 15 FPS in best case for 240p video resolution with a gain of 60% in power consumption.***

Keywords : HEVC; intra prediction; Zynq FPGA.

I. INTRODUCTION

Recently, the Joint Collaborative Team on Video Coding (JCT-VC) [1] group has been working to establish the standardized version of HEVC [2] with the major objective of achieving the best tradeoff between complexity and compression ratio relative to the state-of-the-art H.264/AVC [3]. HEVC provides 36% better efficiency coding and 17% reduction in bit rate for intra prediction case with a loss of 3 times in complexity compared to H.264 [4]. Several new coding tools are adopted in each decoding step presented in Fig. 1. First, a new coding structure based on coding tree bloc (CTB) partition is introduced with three different types of units which are as follow: coding unit (CU), prediction unit (PU) and transform unit (TU). CU is the basic coding unit which can be split from large coding unit (LCU) to small coding unit (SCU). The LCU sizes vary from 8×8 to 64×64 and for the last depth of CU the size of PU is fixed. For luminance component, PU can have the size from 4×4 to 64×64 and it is used for prediction process. Then, depending on PU partition (symmetric/ asymmetric) the size of TU is fixed which is used for transforming and quantization process. The second contribution is highlighted at the level of intra prediction block. In fact, HEVC defines 35 prediction modes including 33 angular prediction modes with DC and planar modes adopted for all PU sizes (4×4, 8×8, 16×16, 32×32 and 64×64). These increasing in computational complexity leads to improve precision during intra prediction process. The final step, after getting the reconstructed frame, consists of applying an anti-bloc filter followed by an additional SAO filter in order to get better picture quality and reduce artifact effects.

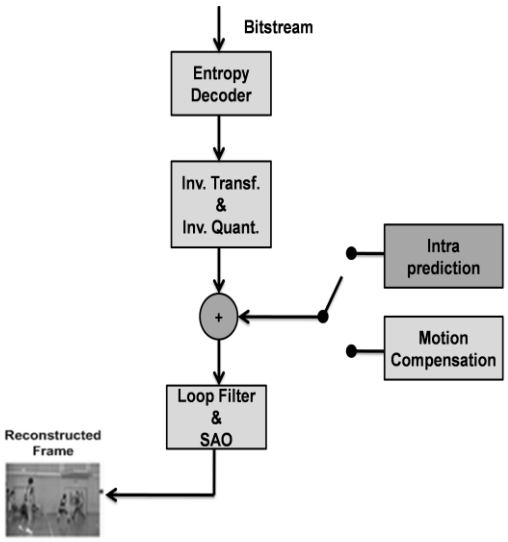


Fig. 1. HEVC decoder layer.

In order to evaluate which part of the decoder chain is the most time consuming, we perform the profiling of the HEVC test model (HM) decoder reference software version 10 [5] using Gprof tool under ARM Linux GCC hosted on Xilinx Zynq ZC702 evaluation board. The profiling is proceeded by decoding 150 frames of "People on Street" video sequence (2560×1600 pixels) for all-intra configuration. We choose four different Quantization Parameter QP values and we keep the default configuration conditions defined in the Common Test Conditions [6]. The profiling result presented in Table I shows the total decoding time on average for all-intra configuration.

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
Manel Kammoun*, LETI Laboratory, University of Sfax, Sfax, Tunisia
Ahmed Ben Atitallah, College of Engineering, Jouf University, Kingdom of Saudi Arabia.

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

When analyzing the statistical provided in Fig. 2, we notice that more than 70% of the total decoder complexity is shared between entropy decoding and
intra prediction modules. In fact, the entropy decoding and the intra prediction consume on average 39% and 32% respectively of the total decoding time.

Table I. Decoding time distribution: All-intra configuration.

Functions	ED [%]	IP [%]	IT/IQ [%]	LF & SAO [%]
QP=22	41	29	14	9
QP=27	39	30	15	6
QP=32	37	35	16	10
QP=37	40	37	16.5	11
Average	39	33	15	9

ED: entropy decoding, excluding. **LF:** Loop filter. **IT/IQ:** inverse quantization and transform. **IP:** intra prediction and picture reconstruction process. **LF & SAO:** Loop filter and sample-adaptive offset filter.

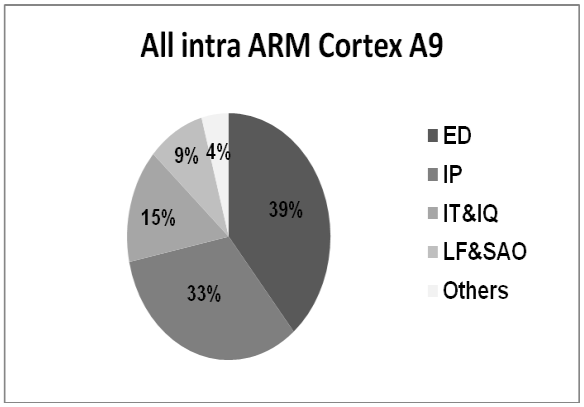


Fig.2. Average decoding time distribution in (%) on ARM Cortex A9 processor for all-intra configuration.

The computational complexity presented in the entropy decoding algorithm makes it difficult to be pipelined and parallelized in such hardware context. This complexity is caused by the critical bin-to-bin data dependencies and the use of fed back between entropy decoding steps [7]. For this reason, we choose to implement in hardware the intra prediction block for PU sizes 4×4 to 16×16 which is the next most complex module. The hardware design is implemented using Xilinx Zynq ZC702 evaluation board and synthesized for TSMC 180 nm CMOS technology. Then, the heterogeneous HW/SW implementation (intra in HW and the rest in SW) of the decoder is validated on the Zynq platform making profit from the multicore ARM and the reconfigurable technology. A particular attention is given to the communication support between the two parts in order to reduce the memory overhead. This paper is organized as follows: In Section 2, we describe the intra prediction algorithm in the HEVC standard. Then, Section 3 illustrates the related works presented in literature. The proposed hardware architecture for prediction units of sizes 4×4 to 16×16 is detailed in Section 4. Moreover, the synthesis results and the comparison studies on Xilinx Zynq based FPGA and 180 nm CMOS standard technologies are detailed in Section 5. In Section 6, we focus on improving the performances of our decoder in term of run-time and power consumption by implementing a HW/SW solution based on the heterogeneous Zynq platform. Finally, the conclusion and future works are provided in Section 7.

II. OVERVIEW OF HEVC INTRA PREDICTION MODULE

In the latest version of HM reference software, intra coding allows 35 prediction modes [8] (33 angular modes + DC + planar) as shown in Fig. 3 (left). For instance, the neighboring pixels used to predict mode 29 are depicted in Fig.3 (right). These modes are specified for 2Nx2N or NxN PU partitions as shown in Fig. 4, where N is the half of CU size. The increased number of prediction modes provides better accuracy to predict complex data in video sequences and also significant reductions in residual energy caused by intra coded frames. However, it involves an increase in computational task comparing to previous standard.

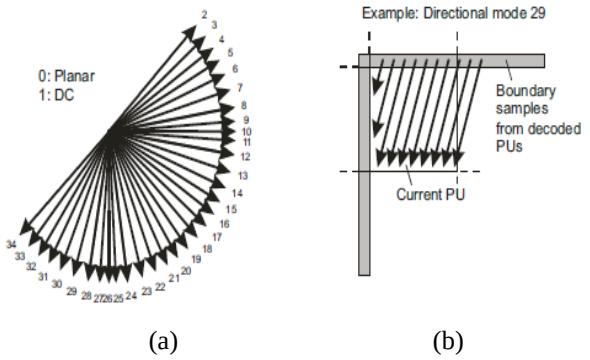


Fig.3. The 35 intra prediction modes (a) and neighboring pixels used to predict mode 29 (b).

Unlike H.264 [9], the prediction process in HEVC requires neighboring information coming from four already coded blocks instead of two blocks. Furthermore, two main references are used for prediction corresponding to the rows of pixels located above and in the left of the current PU. In fact, the main reference used for vertical modes is in the top of the predicted block while the horizontal modes exploit reference pixels situated in the left of the same block. Then, pixels are predicted by using either linear interpolation or copying reference samples depending on integer and fractional parts of interception. In this case, when the fractional part "iFact" of intercepted pixels is not equal to zero, linear interpolation is used for prediction to determine pixel coefficients in the reference array " Idx" and " iFact" given in (1) and (2) then the prediction equation in (3). Otherwise, neighboring pixels are directly copied into the predicted block.

$$Idx = ((y+1) * intraPredAngle) >> 5 \quad (1)$$

$$iFact = ((y+1) * intraPredAngle) \,\&\, 31 \quad (2)$$

$$predSamples[x,y] = ((32 - iFact) * refMain[x + iIdx + 1] + iFact * refMain[x + iIdx + 2] + 16) >> 5 \quad (3)$$

The DC mode uses the average values between reference samples located above and in the left of the predicted block as shown in Fig. 5. There are four different patterns characterizing this mode presented in equation (4), (5), (6) and (7).

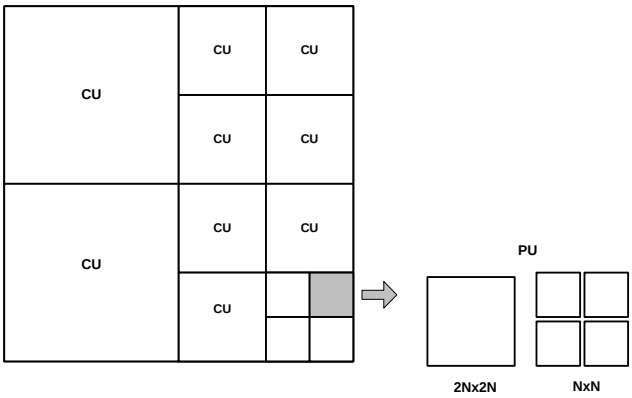


Fig. 4. Extraction intra coded PUs from CUs.

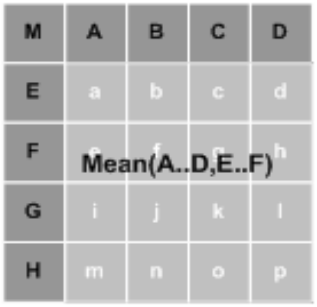


Fig. 5. Intra prediction of the DC mode.

$$DCval = 128 \quad \text{if (None reference)} \tag{4}$$

$$DCval = (A + B + C + D + E + F + G + H + 2) >> 3 \quad \text{if (Top block \& Left block)} \tag{5}$$

$$DCval = (A + B + C + D + 2) >> 2 \quad \text{if (Top block)} \tag{6}$$

$$DCval = (E + F + G + H + 2) >> 2 \quad \text{if (Left block)} \tag{7}$$

III. RELATED WORKS

Since the emergence of the first working drafts in 2010, several researches have been devoted to enhancing intra prediction techniques compared to the previous standards with a main motivation to simplify the implementation of this algorithm. In [10], Li *et al.* proposed a new flexible selection technique of reference pixels for 4×4 intra prediction which can save significantly the hardware cost. However, it can support only 17 prediction modes defined in HM 3.0 instead of 35 modes. In addition, this architecture requires 24 clock cycles with 8 clocks processing latency to perform one prediction mode of 4×4 PU which can affect the speed-up of the design. Ercan *et al.* [11,12] exploit in their work data reuse and pixel equality techniques already used in H.264 [13] in order to reduce the power consumption and the area cost drastically. This design can proceed the prediction of 5 pixels in parallel of four 4×4 PU and one 8×8 PU and it can process 30 full HD (1920x1080) video frames per second. The implementation is done on Virtex 6 FPGA and verified to run at 166.7 MHz. Despite the statistical analysis mentioned in [10] indicating that 99% of computed blocks is shared between 4×4, 8×8 and 16×16 blocks, the hardware architecture presented in [14] computes all PU sizes with 64 parallel processing elements able to support all modes. This excessive use of parallel process has a real impact on the area cost as well as the power consumption. Furthermore, the proposed design can process the decoding of only 60 frames per second for full HD (1920x1080) resolution. As well, the hardware approach detailed in [15] performs all the PUs with an operating frequencies 100 MHz executed on Altera Aria II GX FPGA. The processing time spent for 32×32 PU is relatively important (between 3305 and 1545 clock cycles) which enables the process of 15 to 30 frames per second for full HD sequence. Finally, authors in [16] present an efficient hardware architecture dedicated for 4×4 PU which exploits the symmetric proprieties vertical and horizontal modes. In this work, two prediction units are running in parallel to proceed 32 pixels at the same time which leads to improve the maximal throughput required to process real time decoding of full HD resolution. Unfortunately, the excessive use of parallelism can affect the power consumption and increase latency time especially for memory access overhead and data distribution. In this work we will try in the next section to propose an efficient solution that can offer a good trade-off between performance, area utilization and power consumption.

IV. PROPOSED HARDWARE ARCHITECTURE

Since the prediction units of sizes 4×4, 8×8 and 16×16 are the most computed in such HD video sequence [10], it would be wise to design a new hardware architecture that supports 4×4, 8×8 and 16×16 PUs. Our proposed architecture is shown in Fig. 6. It consists of a sample prediction engine that performs the prediction of 16 pixels in parallel, a control unit to regulate the data flow and a bloc register to store the predicted pixels of different prediction modes. In this work, 33 angular and the DC modes are proceeded, the planar mode is not considered. Furthermore, we exploit the same symmetric propriety already used in [16] in order to reduce the number of computed modes. In fact, when analyzing the prediction modes for both horizontal and vertical directions, we have noticed that the corresponding prediction equations are symmetric and only selected pixels of extended main references are modified depending on the prediction direction. Hence, it is possible to compute prediction modes of one direction by switching between the main references of vertical and horizontal directions. For example, the number of equations calculated for 4×4 PU luminance components will be equal to 224 equations instead of 448.This solution can help to reduce the computational complexity presented in this module and also to improve the throughput required for achieving real time decoding of Ultra-HD video frame resolution. There are three possible cases to get the predicted samples: the first case is chosen when the fractional part of intercepted pixels is equal to zero which corresponds to diagonal modes or intra prediction angle is equal to zero.

In this case, reference samples are transmitted directly to the outputs of the design without any computational tasks. The second case is selected when interpolation is used for prediction (fractional part is not equal to zero). Therefore, we introduce in the input of the design a multiplexer "MUX" for selecting the reference samples used to operate one prediction equation. Then, pixels of the reference array are incremented until processing all the block size. During this stage, the sample operative engine calculates the predicted pixels for each pair of reference samples and then the output values are stored in block registers. The final step consists of using another "MUX" in the output to select the appropriate predicted pixels corresponding to the decision mode (chosen by the encoder) as shown in Fig. 6. Finally, the third case is computed when none directional mode is selected (DC). Therefore, we have to calculate the average values of available reference samples as given in equations (8), (9), (10) and (11).

$$DC0 = 128 \quad \text{if (None reference)} \tag{8}$$

$$DC1 = \left(\sum ref.top[i] + \sum ref.left[i] + 2\right) >> 3 \quad \text{if (Top block \& Left block)} \tag{9}$$

$$DC2 = \left(\sum ref.top[i] + 2\right) >> 2 \quad \text{if (Top block)} \tag{10}$$

$$DC3 = \left(\sum ref.left[i] + 2\right) >> 2 \quad \text{if (Left block)} \tag{11}$$

The general structure of the sample prediction engine is illustrated in Fig. 7. It can perform the prediction of any modes and any PU size which helps to exploit better correlations between different modes and reduces hardware cost. During the processing of the prediction engine, 16 predicted pixels are proceeded in parallel. We need to select two pair of reference samples to proceed all the possible prediction equations in each clock cycle. When performing intra prediction 4×4, the width of inputs used may be equal to 8×8bits which is the largest data size defined for one prediction mode. However, intra prediction 8×8 and 16×16 use a maximum input data size of 15×8bits and 32×8bits respectively.

The proposed hardware architectures need at most 60 clock cycles to display all prediction modes (33 angular modes +DC) for intra-16×16, 30 clock cycles for intra-8×8 and 15 clock cycles to perform hardware acceleration of intra-4×4 as shown in Fig. 8. In fact, 2 cycles of latency are necessary to start generating 16 predicted pixels in each clock cycle which help to reach maximum throughput of Ultra HD video resolutions.

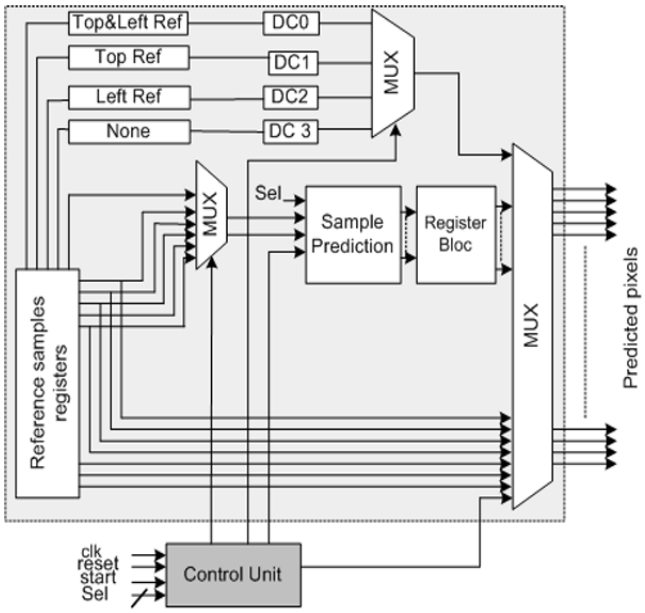


Fig. 6. Proposed hardware architecture.

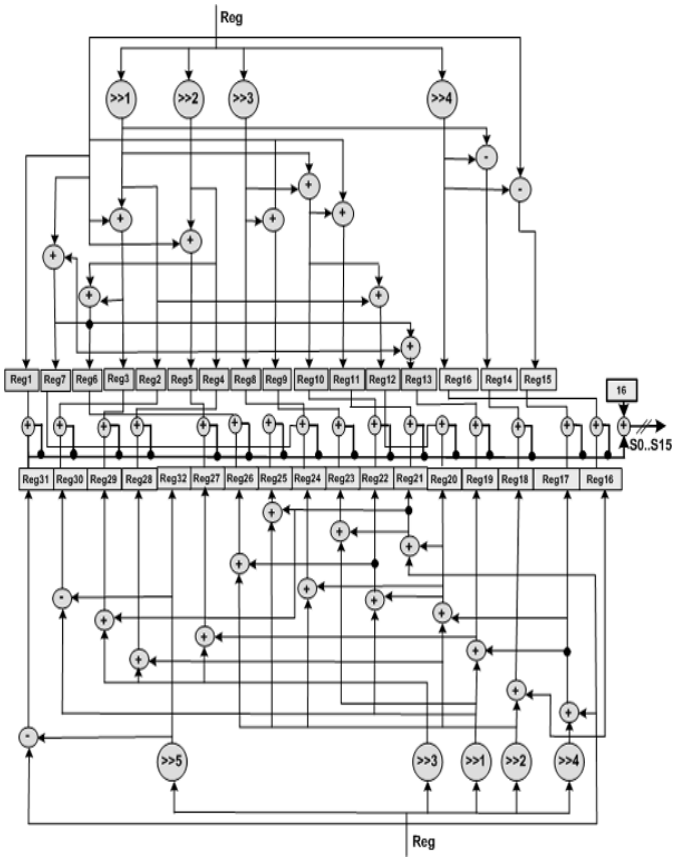


Fig. 7. The sample prediction engine.

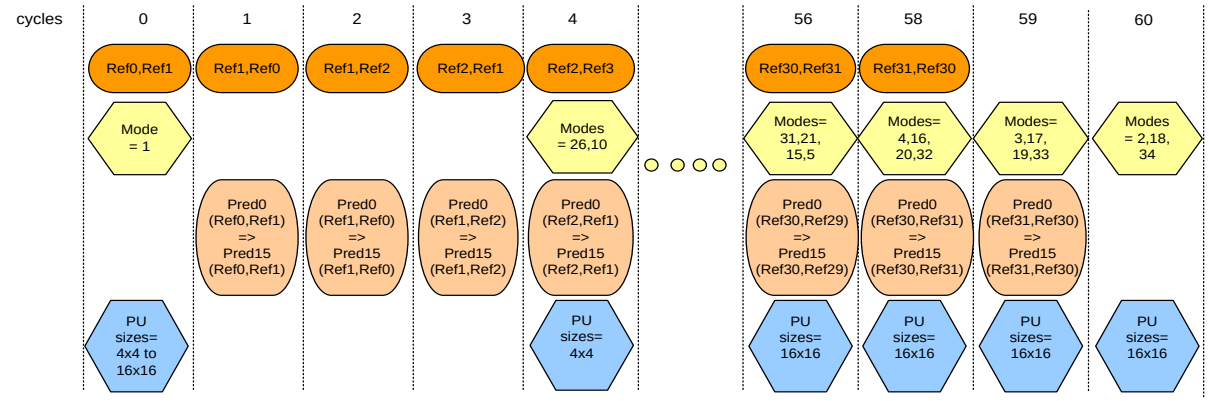


Fig. 8. The number of clock cycles required for 4x4, 8x8 and 16x16 PU sizes.

V. IMPLEMENTATION RESULTS ON FPGA AND ASIC TECHNOLOGIES

The proposed hardware architecture is designed with VHDL language, simulated by Mentor Graphics ModelSim simulator and synthesized with Loenardo Spectrum using TSMC 180 nm CMOS and Xilinx XC7Z020 FPGA Xilinx ISE 14.7 tools. Table II illustrates a comparison study between our approach and previous works.

According to the results supplied in Table II, we can notice that the throughput reached by the intra-4×4 hardware architecture can process the decoding of 4096×2048@45 video frames per second instead 1920×1080@45 frames per second in cost of increasing the hardware area with about 36 % compared to [10]. Indeed, the operating frequency is equal to 222 MHz with an improvement by 7% in total gate count compared to [16]. On the other hand, the synthesis results given by intra-8×8 hardware design show an increase in area cost with about 50% relative to intra-4×4 while the operating frequency is almost the same. In case of intra prediction 16×16, the total gate count is equal to 99K gates and the operation frequency is about 205 MHz.

When estimating the performances of our designs relative to other previous works, it seems that the throughputs in [14] and [15] can only process the decoding of 29.4 and 60 frames per second of 1920×1080 video resolution respectively. In return, we are able in our cases to achieve the decoding of 28 frames per second at least for Ultra-HD resolution (4096×2048). This feature is back to the choice of blocks implementation. In fact, we choose to implement PU sizes independently in order to avoid data dependency between blocks and save processing time. Moreover, the design proposed in [14] exploits 64 processing units (PEs) working in parallel inducing an increasing of 3 times in area cost compared to intra-16×16 architecture. Taking into account that 99% of computed blocks is shared between 4×4, 8×8 and 16×16 [10], there is no need to proceed in hardware large PU sizes.

Table III reports the synthesis results on Xilinx Zynq 7020 FPGA. All FPGA implementations are verified to work at 232 MHz under XC7Z020 FPGA device. The intra-8×8 implementation proves an increase with about 50% in the number of used slices compared to [11]. Nevertheless, the hardware cost of intra-4×4 is reduced by 1% compared to [16].

Table II. Comparison results with ASIC technology.

Architectures	**[10]**	**[14]**	**[15]**	**[16]**	**Proposed Intra 4×4**	**Proposed Intra 8×8**	**Proposed Intra 16×16**
Technologies (nm)	TSMC 130	TSMC 130	TSMC 130	TSMC 180	TSMC 180	TSMC 180	TSMC 180
Frequencies (MHz)	150	400	200	218	222	217	205
Area (Kgates)	9	324	75.5	15	14	31	99
Total modes	1	35	35	34	34	34	34
PU sizes	4×4	All PUs	All PUs	4×4	4×4	8×8	16×16
Processing latency (clocks)	8	-	-	2	2	2	2
Throughput Fps	1920×1080 @45	1920×1080 @60	1920×1080 @29.4	4096×2048 @45	4096×2048 @28	4096×2048 @55	4096×2048 @104

Table III. Comparison results with FPGA technology.

Architectures	**[11]**	**[15]**	**[16]**	**Proposed Intra 8×8 on Virtex 6**	**Proposed Intra 4×4 on Zynq**	**Proposed Intra 8×8 on Zynq**	**Proposed Intra 16×16 on Zynq**
Technologies	Xilinx Virtex 6	Altera Aria II GX	Xilinx Zynq	Xilinx Virtex 6	Xilinx Zynq	Xilinx Zynq	Xilinx Zynq

Area cost	4835 LUTs (5%)	20496 ALUTs	2110 LUTs (4%)	4747 LUTs (10%)	1668 LUTs (3%)	4762 LUTs (9%)	16486 LUTs (31%)
Frequencies (MHz)	166.7	100	229	238	232	232	232
Total modes	34	35	34	34	34	34	34
Throughputs Fps	-	1920×1080 @ 31.1 (without 4×4 PUs)	45 4096×2048	80 4096×2048	28 4096×2048	80 4096×2048	118 4096×2048
PU sizes	8×8	All PUs	4×4	8×8	4×4	8×8	16×16

VI. HW/SW Implementation And Performances Evaluation

A. Platform features

The heterogeneous HW/SW implementation is carried on Zynq 7020 SoC platform [17][18]. It is an example of such circuit integrating a processing system (PS) based on a dual core ARM Cortex A9 processor and a programmable logic (PL) of FPGA series 7. Inside the Zynq architecture, there are different communication supports to connect the processor with the reconfigurable logic. Xilinx adopted the Advanced extensible Interface AXI4 of Advanced Microcontroller Bus Architecture (AMBA) [19] protocols as the next-generation IP interconnect standard for FPGA Series 7. Hence, the types of protocols available in this platform are as follows: four High performance AXI4 ports 32/64 bits (two slaves and two masters), four general purpose port 32 bits (two slave and two mater) and one 64 bits Accelerator coherency (ACP) port. They are used to exchange data between PS and PL with an efficient and flexible way as shown in Fig. 9.

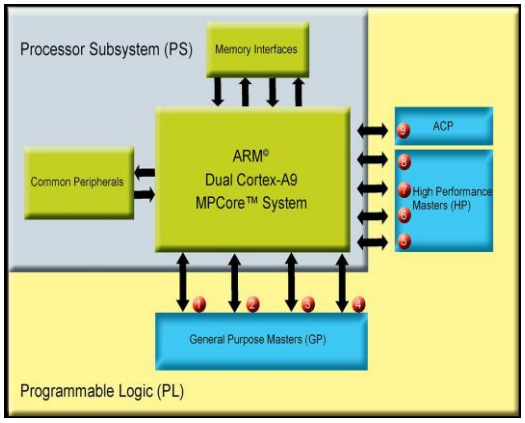


Fig. 9. Zynq communication protocols.

The AXI4-lite interface is designed for serial communication with address and control register of size 32 bits. However, the AXI4-stream provides a high bandwidth and a direct access to the On-Chip Memory (OCM) and DDR memory through the Direct Memory Access (DMA) channel. This mode of transfer can support illumined size of data and can provide point to point streaming interface without using any address.

In our work, we will limit the study to the streaming interface the fact that it seems to provide better performances in run-time and power consumption compared to the AXI4-lite interface. In fact, as the complexity of hardware IP increases, as the disadvantages of the serial transfer are more and more proved. The performance evaluation of this study is provided in the next sub-section.

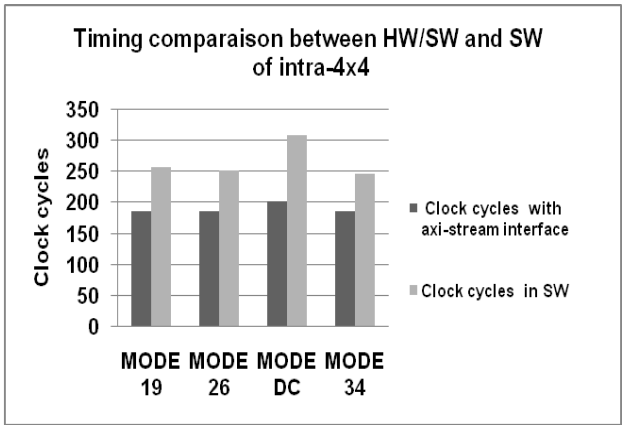
B. Performances evaluation

The hardware IPs are created and customized using Vivado Design Suit while SW descriptions are compiled with standalone application on the ARM processor using software development kit tool (SDK) in windows development environment. We use for verification the test data of 2560×1600p encoded video sequence. Table IV illustrates the amount of reconfigurable logic area used for mapping LUT, Block RAM and Flip-Flop blocks for the streaming mode using Vivado 2015.4 tool.

Table IV. FPGA implementation results with streaming interface.

Architectures	**Proposed Intra 4×4**	**Proposed Intra 8×8**	**Proposed Intra 16×16**
Technology	XC7Z020 FPGA	XC7Z020 FPGA	XC7Z020 FPGA
Number of slice (LUTS)	4882 (9%)	8033 (15%)	22713 (42%)
Frequencies (MHz)	232	232	232
Number of Block RAM/FIFO	5 (3%)	5 (3%)	5 (3%)

In the next task, the timer incorporated in the ARM processor is exploited to accurately measure the average decoding time for the HW/SW design using "xtime l.h" library. The timing analysis of HW/SW implementation of 4×4 PU size for some prediction modes with AXI4-stream interface is detailed in Fig. 10. We choose to compare the run time of 4 prediction modes which are one angular mode (mode 19), DC, diagonal (mode 34) and copying mode (mode 26).



zFig. 10. Timing performance analyze of intra-4×4.

Retrieval Number: F2440037619/19©BEIESP
DOI: 10.35940/ijrte.F2440.078219
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

From Fig. 10, we can see that the total clock cycles spent by HW/SW implementation with streaming interface is improved by 30% to 40% compared to the SW case. In the next step, we perform the same comparison study with intra prediction 8×8 and 16×16. The results of these measurements are provided in Fig. 11 and Fig. 12 respectively. These analyses confirm the results given by intra 4×4. In fact, we can notice that the run-time of HW/SW designs is always less than SW approach whatever is the degree of hardware complexity. In these cases, the gain is more than 50% relative to SW which demonstrates the efficiency of the AXI4-stream interface for highly communicating tasks.

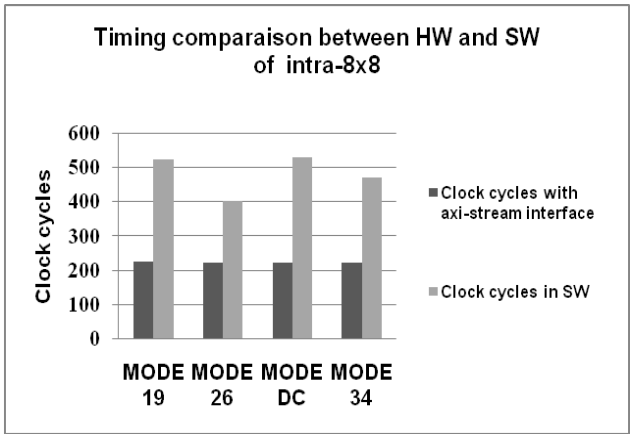


Fig. 11. Timing performance analyze of intra-8×8.

The next stage of the project consists in testing the full HEVC decoder for different video classes. To complete this task, four input bitstream files are used corresponding to video classes A, B, C, and D as well as four quantization steps equal to 22, 27, 32 and 37. In Fig. 13, the average value is computed of frame rates corresponding to four quantization steps. From this study, it's evident that the HW/SW design can achieve a gain with about 3% in total decoded frames compared to SW solution.

At last, we connect the USB interface adapter of Texas Instruments to the Zynq board. Then, we calculate the power consumption for both implementations (heterogeneous HW/SW and totally SW) using Xilinx Fusion Digital Power designer tool. Table V illustrate the consumed power for 4×4, 8×8 and 16×16 intra prediction blocks using HW/SW and SW implementations as well for Full HEVC decoder.

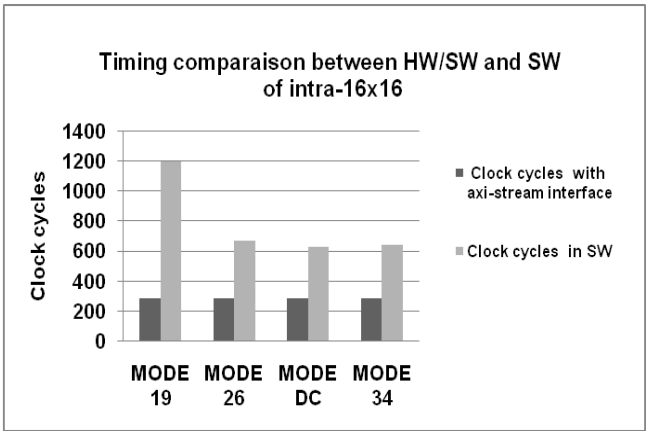


Fig. 12. Timing performance analyze of intra-16×16.

Fig. 13. Average frame rate (FPS) for SW and HW/SW using different video classes.

It is evident from Table V that the power consumptions of HW/SW with streaming interface are less by nearly 80 % than SW consumption for the different proposed architectures. In addition, the full HW/SW decoder records a gain with about 60% compared to SW implementation. Finally, we can notice that the variation of power consumption between HW/SW cases is proportional to the amount of time consumed during transferring data.

Table V. Power estimation of SW and HW/SW implementations.

Intra-4×4		
Implementations	**SW**	**HW/SW**
Power consumption (W)	0.38	0.063
Intra-8×8		
Implementations	**SW**	**HW/SW**
Power consumption (W)	0.39	0.07
Intra-16×16		
Implementations	**SW**	**HW/SW**
Power consumption (W)	0.4	0.074
Full HEVC decoder		
Implementations	**SW**	**HW/SW**
Power consumption (W)	0.423	0.157

To summarize, during these previous experiments we emphasized the intake of using hardware acceleration on processing time and power consumption especially when the computational complexity of algorithms is important.

VII. CONCLUSION

In this paper, we proposed an efficient hardware architecture for HEVC intra prediction decoder including PU of sizes 4×4, 8×8 and 16×16. It exploits the symmetric property between vertical and horizontal modes which leads to save the hardware cost and improve the processing time of the programmable logic. The operation frequencies are about 205 MHz and 232 MHz respectively with ASIC and FPGA technologies. Furthermore, we proposed in this work a heterogeneous HW/SW implementation that uses the ARM Cortex A9 processor and the programmable logic in order to estimate the run-time and the power consumption parameters.

The experimental results show an improvement between 30% to more than 50% in processing time for the proposed hardware architecture. Indeed, the power consumption is saved with about 80% compared to the SW implementation. Otherwise, the full HW/SW decoder provides an improvement with about 3% in total decoding time while the power consumption is 60% better than SW case. As future works, we will introduce partial reconfiguration in our design to implement dynamically the different PU sizes which helps to reduce better area utilization and power consumption.

REFERENCES

1. JCT-VC, http://www.itu.int/en/ITU-T/studygroups/2013 -2016/16/ Pages/ video/ jctvc.aspx.
2. G. J. Sullivan, J-R. Ohm, w.-J. Han, J. Min, and T. Wiegand,*"Overview of the high efficiency video coding (HEVC) standard"*, IEEE Trans. Circuits Syst. Video Technol. , vol,. 22,no. 12pp. 1792-1801, Dec 2012.
3. A. Ben Atitallah, H. Loukil, P. Kadionik, N. Masmoudi "*Advanced design of TQ/IQT component for H.264/AVC based on SoPC validation*", WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS, Vol. 11, Issue 7, July 2012.
4. S. Park, J. Park, and B. Joen, "*Report on the evaluation of HM versus JM,"* JCTVC-D181, Jan. 2011.
5. 5. B. Bross, F. HHI, W. Jin Han, .all, , *"High Efficiency Video Coding (HEVC) text specification draft 10"*, JCTVC-L1003_v9, 12th Meeting: Geneva, CH, 14–23 Jan 2013.
6. 6 F. Bossen, *"HM 8 Common Test Conditions and Software Reference Configurations"*, JCTVC-K1100, JCT-VC, China, Oct. 2012.
7. Z. Dajiang, Z. Jinjia, F. Weib and G. Satoshi, *"Ultra-high-throughput VLSI Architecture of H.265/HEVC CABAC Encoder for UHDTV Applications"*, IEEE Transactions on Circuits and Systems for Video Technology, July 2014.
8. 8. Lainema, J.; Bossen, F.; Woo-Jin Han; Junghye Min; Ugur, K., *"Intra Coding of the HEVC Standard"*, Circuits and Systems for Video Technology, IEEE Transactions on , vol. 22, no.12, pp. 1792, 1801, Dec. 2012.
9. S. Smaoui, H. Loukil, A. Ben Atitallah, N. Masmoudi, *"An Efficient Pipeline Execution of H.264/AVC Intra 4x4 Frame Design "*, IEEE SSD'10, Amman, Jordan, 27-30 June 2010.
10. F. Li, G.Shi1, F.Wu, *"An efficient VLSI architecture for 4×4 intra prediction in the high efficiency video coding (HEVC) standard"*, 18th IEEE international Conference on Image processing, China 2011.
11. E. Kalali, Y.Adibelli, I. Hamzaoglu, *"A high performance and low energy intra prediction hardwae for high efficiency video coding "*, Field Programmable Logic and Applications (FPL), 2012.
12. E. Kalali, Y.Adibelli, I. Hamzaoglu, *"A low energy intra prediction hardware for high efficiency video coding"*, Journal Real-Time Image Processing, DOI 10.1007/s11554-014-0471-5 , pp 1-14, 31 December 2014.
13. Y. Adibelli, M. Parlak, I. Hamzaoglu, *"Computation and Power Reduction Techniques for H.264 Intra Prediction"*, Microprocessors and Microsystems: Embedded Hardware Design, vol. 36, issue 3, May 2012.
14. N. Zhou, D. Ding, Y. Lu, *"On hardware architecture and processing order of HEVC intra prediction module"*, Picture Coding Symposium, San Jose, CA, USA, December 2013.
15. A. Abramowski, G.Pastuszak, *"A novel intra prediction architecture for the hardware HEVC encoder"*, 16th Euromicro Conference on Digital System Design, Los Alamitos, CA, USA, 4-6 Sept. 2013.
16. M. Kammoun, A. Ben Atitallah and N. Masmoudi, *"An optimized hardware architecture for intra prediction for HEVC"*, International Image Processing, Applications and Systems Conference, Sfax, Tunisia, 5-7 Nov. 2014.
17. Xilinx, Zynq-7000 All Programmable SoC Software DevelopersGuide, Version 9.0, June 4, 2014.
18. http://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf.
19. M. Bouain, R. Ben Atitallah, A. Ben Atitallah, Nouri Masmoudi, J. L. Dekeyser "*Design Space Exploration on Heterogeneous SoC: The H.264 encoder case-study*", GDR SOC-SIP'13, Lyon, France, 10-12 June 2013.
20. ARM. Inc. AMBA AXI and ACE Protocol Specification, Feb. 2013 http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022e/index.html.

AUTHORS PROFILE

Manel Kammoun Received here PHD degree in Electronics from the National School of Engineering-Sfax (ENIS) in 2018 and the diploma of engineer in Electronics from the University of Sfax in 2012. She is currently researcher in the Laboratory of Electronics and Information Technology whitin the C&S (Circuit & System) team. Here main research activities are focused on image and video signal processing, hardware implementation and embedded systems.

Ahmed Ben Atitallah is currently an Associate Professor in Electronics at College of Engineering, Jouf University, Kingdom of Saudi Arabia. He received his PhD degree in Electronics from the University of Bordeaux1 in 2007 and the diploma of engineer and MS degree in Electronics from the University of Sfax in 2002 and 2003, respectively. From 2008 to 2018, he held the position of assistant professor then associate professor in Electronics at the University of Sfax. His main research activities are focused on image and video signal processing, FPGA implementation, embedded system design.