Mohamed, Belal; Shalaby, Ahmed; Sayed, Mohammed S.

Conference Paper

High-Level Synthesis Hardware Accelerators of Integer-pixel Motion Estimation of HEVC on SoC-FPGA Platform

2nd Europe - Middle East - North African Regional Conference of the International Telecommunications Society (ITS): "Leveraging Technologies For Growth", Aswan, Egypt, 18th-21st February, 2019

Provided in Cooperation with:

International Telecommunications Society (ITS)

High-Level Synthesis Hardware Accelerators of Integer-pixel Motion Estimation of HEVC on SoC-FPGA Platform

Belal Mohamed, Ahmed Shalaby, and Mohammed S. Sayed

Abstract

Motion estimation entails the major computation complexity load and processing time in HEVC video encoder. Integer-pixel Motion Estimation (IME) consume more than 45% of the processing time. Therefore, this paper presents a High-Level Synthesis Hardware Accelerator for Integer-pixel Motion Estimation of HEVC on Xilinx SoC-FPGA Platform. The hardware accelerator is three time faster than the corresponding software implementation with only 100 MHz clock frequency on Xilinx Zynq ZC702 FPGA.

I. Introduction

The new H.265/HEVC video coding standards [1] added huge computational complexity to achieve double the compression ratio (i.e. 50% bit rate reduction) for the same video quality in comparison with the preceding H.264/AVC standard. This high computational cost represents a big challenge for real-time compression of video data. Integer-pixel and fractional pixel motion estimation, DCT and quantization, and intra-prediction represent almost 90% of the processing load of the HEVC encoder. Processing time profiling was carried in [2]. Based on the profiling results, Motion estimation with its two parts; integer-pixel and fractional pixel, accounts for more than 65% of the processing time (i.e. processing load), DCT and quantization consumes about 15% of the processing time, intra-prediction consumes about 1.5% of the processing time, and memory setting and copying consumes about 7% of the processing time.

II. Architecture

Motion estimation (ME) is used for inter frame prediction where the motion in the current frame is predicted from one or more reference frames by calculating a motion vector for every block. In our implementation, Integer-pixel Motion Estimation (IME) computes the Integer Motion Vector (IMV) by utilizing full search algorithm within a search range ±16 pixels. As shown in Figure 1(a), the architecture is composed of 1) the input memory of the search window, 2) the input memory of the reference block, 3) Sum of Absolute Difference (SAD) unit, 4) comparison unit and 5)

motion vector calculator unit. Outputs of the IME are minimum cost value and Integer motion vector, which are saved in registers.

- **Memory Organization**

The block size in our implementation is fixed to the size of 16 x 16 pixels and the pixel is placed in 8-bits width. The search window size is configured as 48 x 384 = (32 + 16) x (48 * 8 bits), while the reference block size is configured as 16 x128 = 16 x (16 * 8 bits). The search window memory is filled in 576 cycles. The writing process is shown in Figure 1(b), where four pixels are sent to the search memory at a cycle. Each row is a register of size 384 bits (48 pixels). In 12 cycles, the registers is filled then the write process move to the next row. The total number of cycles is 576 (12 cycle per row x 48 rows). While the reference block memory is filled in 64 cycles. Likewise, four pixels are written to the memory at a cycle. Each row is a register of size 128 bits (16 pixels). In 4 cycles, the registers is filled then the write process move to the next row. The total number of cycles is 64 (4 cycle per row x 16 rows).

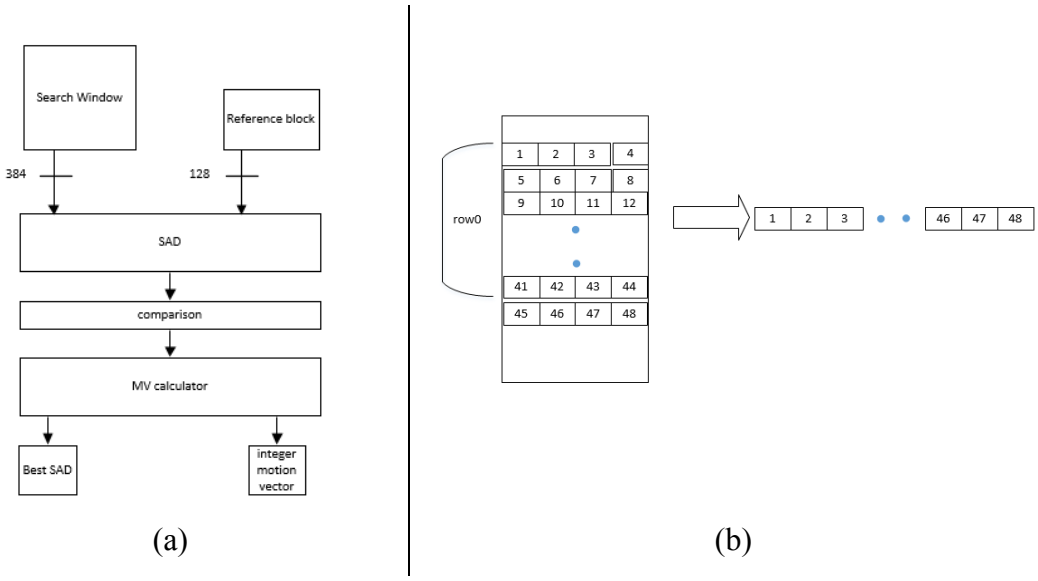

Figure 1 a) Block diagram of Integer-motion estimation b) The search window write process

- **Processing Unit**

The processing unit is divided into three units [3]: 1) SAD unit shown in Figure 2(a), 2) Comparison unit, and 3) motion vector calculator unit. SAD unit receives the input values of the pixels from the search window memory and reference block memory. Then the Processing Element (PE) works to calculate the SAD value between the pixels and its references. Next, the

output values is stored at the output SAD register. The architecture of the SAD unit is shown in Figure 2(a). A pipelined architecture of the PE is developed in order to speed up the processing time, as shown in Figure 2(b). Thirty-three PEs work in parallel to calculate SAD values for thirty-three overlapped blocks of a row. Finally, for the thirty-three SAD values, the minimum value and its index are computed by the comparison and MVC units. Since our implementation utilizes the full search algorithm, this process is repeated for all blocks till the integer motion vector and its corresponding minimum SAD value are found and stored.

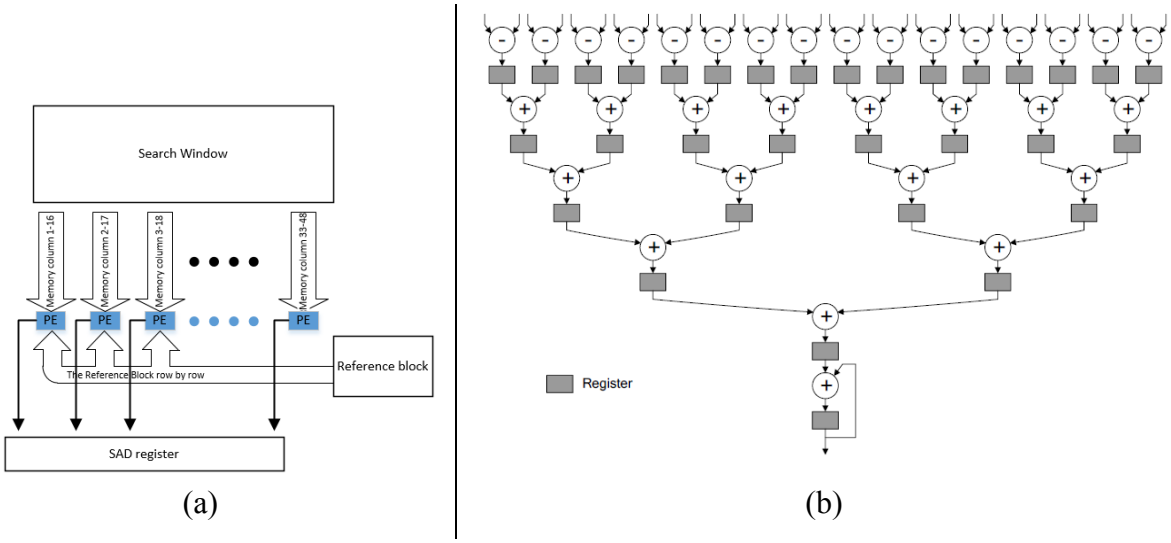

Figure 2. a) SAD unit Architecture b) Processing Element (PE) of the SAD unit

III. Development and Resource Utilization

Hardware/Software co-design aims to exploit the interaction between hardware and software with the goal to optimize and/or satisfy the design metrics such as cost, performance, power, and time-to-market. In this context, System-on-Chip FPGAs platforms are produced to accelerate the development of complex electronics systems based on HW/SW co-design methodology. SoC-FPGAs are programmable System-on-Chip (SoC) that combines programmable logic and microprocessor unit to implement the hardware and software respectively. In this paper, HEVC encoder, Kvazaar [4], is ported to run on the processing unit (i.e. microprocessor) inside the FPGA and the hardware core IP for IME is implemented by High-Level Synthesis HLS approach using Vivado [5]. The IME core is implemented and verified using real-time test vectors from Kvazaar. Our platform is Xilinx Zynq ZC702, evaluation kit in [6].

The proposed architecture of Integer Motion Estimation (IME) was developed using High level Synthesis (HLS) technique on Vivado platform from Xilinx. Former, C-code of the IME algorithm is optimized to be hardware friendly. Latter, the next step is to design the interfaces and apply the proposed architecture. The Interface with the IME core is 32-bits AXI-bus where data is transmitted and received through it. Four pixels are sent to the core per a clock cycle to fill the memory over the AXI-bus. The IME core is synthesized on Xilinx Zynq ZC702 FPGA. The resource utilization of the implemented core is illustrated in Table 1.

Table 1. Summary of resources utilization of IME core on Xilinx Zynq ZC702 FPGA

Summary

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	-	-
FIFO	-	-	-	-
Instance	0	-	8526	29630
Memory	15	-	1056	297
Multiplexer	-	-	-	83
Register	-	-	7	-
Total	15	0	9589	30010
Available	280	220	106400	53200
Utilization (%)	5	0	9	56

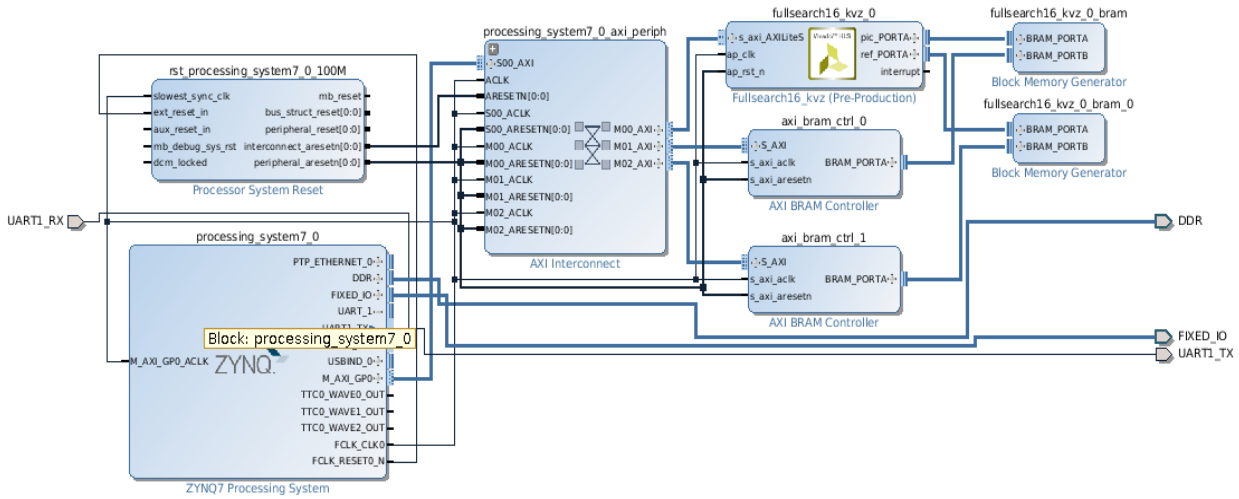

Figure 3. Proposed HLS Architecture for the IME Core “fullsearch16_kvz_0”

IV. Prototyping and Integration

The proposed IME hardware accelerator is integrated with Kvazaar HEVC encoder software, where Kvazaar is running on the ARM processor. Figure 3 shows the details of the integration. The component of the system are 1) the processor, in Zynq ZC702, which is a Dual ARM Cortex-A9 core processor. 2) The generated IME core by HLS. 3) BRAMs to hold the data between the IME core and the processor. 4) Advanced Extensible Interface AXI Bus to connect between the processor, BRAMs and IME core.

V. Testing Methodology

The developed IME hardware accelerator is tested on Xilinx Zynq ZC702. Our aim is to verify IME module functionality using real-time test vectors. Also, to discover the limitations regarding the time of the target platform. In the verification test, the IME module is implemented as an external peripheral on FPGA logic then it is feed by test-vectors from the software running on the processor. Kvazaar software is ported as an application on Embedded Linux. In order to perform this task, the following steps are required:

1. IME module is implemented by standalone-BSP Board Supported Package methodology [7]. Standalone-BSP is a simple, low-level software layer, it provides access to basic processor feature such as cashes, interrupts, exceptions as well as the basic features of hosted environment such as standard input and output, abort and exit. Using standalone-BSP, IME module is implemented as an external peripheral for Cortex-A9 with fixed physical address added to Cortex-A9 memory mapping table. Whenever the Cortex-A9 wants to communicate with the IME module, it just calls it as a hardware peripheral like any external peripheral USB, I2C, SPI, etc.

2. Kvazaar is ported on ARM Cortex-A9 running on Embedded Linux using PetaLinux. The PetaLinux tools offer everything necessary to customize, build and deploy Embedded Linux solutions on Xilinx processing systems [8]. In this context, Kvazaar is cross-compiled, using GNU tool, as an application for Embedded Linux. However, Kvazaar is modified to call the IME hardware module instead of calling module function in the software flow.

3. The image of the complete solution including the hardware description (standalone BSP IP Module) and the Embedded Linux application is built and uploaded to SD card, where The

First Stage Boot Loader (FSBL) in the Zynq ROM reads the "boot.bin" file from the SD card and start the Kvazaar application.

This method is utilized for verification to drive the IME module with real-time test vectors. In the following section, we present the results of the verification test. Screenshots from the test are displayed. Also the timing results are demonstrated. In addition, a comparison between the software IP and the HLS accelerators are showed to demonstrate the gain achieved by integrating IP modules as Hardware on programmable logic.

VI. Integer-pixel motion estimation architecture's test:

In our implementation, Integer Motion Estimation (IME) computes the Integer Motion Vector (IMV) by utilizing full search algorithm within a search range ±16 pixels. In our testcase, each pixel is stored as an integer. So for 16 × 16 block size, the input test vector is 64 pixels × 32 bits. The test vector is generated by Kvazaar then feed to the IME module. Next, the test runs and the output is stored and compared with the reference output. The output vector contains two integer, IMV and best SAD values.

- **Test Screenshot**

In Figure 4, a screenshot of the test is captured. The test is passed as shown in the figure, the done signal is triggered after all output vector is compared and matched to the reference output. The yellow marker shows the end of the testcase.

- **Timing Report**

In order to verify the benefits of the HLS flow, a timing analysis is performed to compare between the pure software IME module runs on ARM processor and HLS IP module. Table 2 shows the timing report of IME. The IME HLS IP module runs at 100 MHz clock, with 1826 clock interval. The output is produced after 1826 / 100 MHz around 18.26 μsec. while in software, it takes 54.5 μsec. So HLS IP module accelerate around three times over the software IP.

VII. Acknowledgment

We would like to thank Egypt-Japan University of Science and Technology (E-JUST) for the continuous support and the National Telecom Regulatory Authority (NTRA) of Egypt for funding this work.

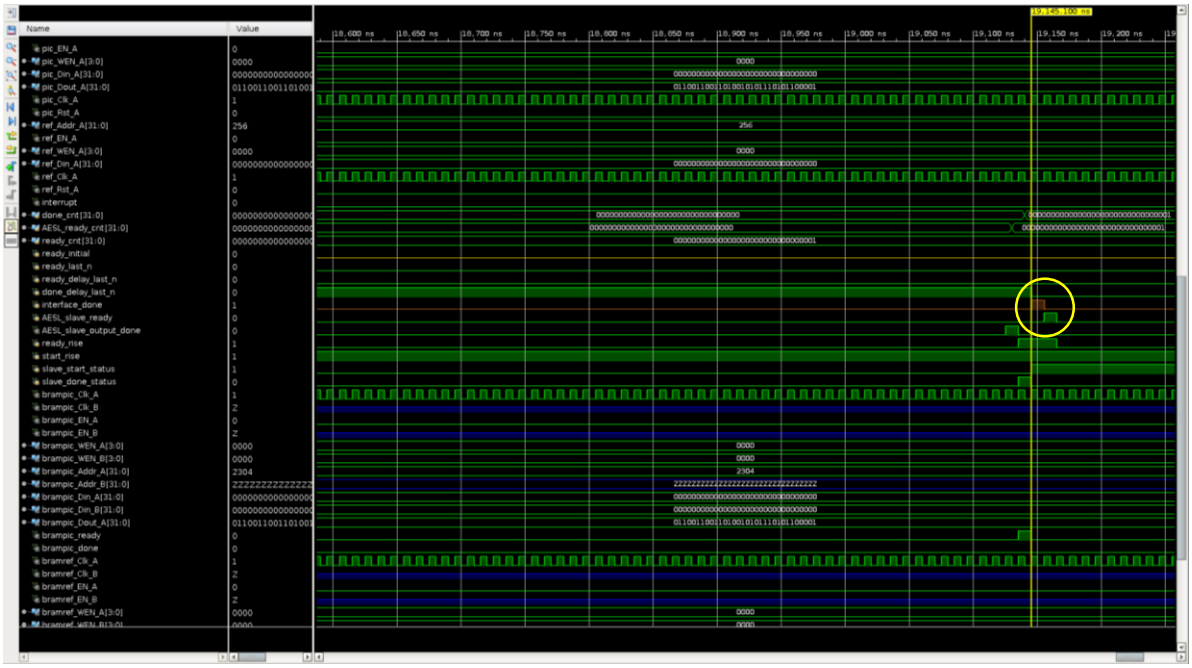

Figure 4. Test Screenshot of Integer Motion Estimation IP Module

Table 2. Timing Report of Integer Motion Estimation IP Module

fullsearch16_kvz_csynth.rpt

Synthesis Report for 'fullsearch16_kvz'

General Information

Date:	Tue May 15 13:04:30 2018
Version:	2014.4 (Build 1071461 on Tue Nov 18 16:42:57 PM 2014)
Project:	sadFull16
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
default	10.00	8.69	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
1825	1825	1826	1826	none

VIII. References:

[1] Kim, I.-K., et al.: High efficiency video coding (HEVC) test model 10 (HM10) encoder description. Doc.JCTVC-L1002. 12thMeeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Geneva, CH (2013).

[2] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC Complexity and Implementation Analysis," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1685-1696, Dec. 2012.

[3] 51. M. Sayed, "A Fast Architecture for Exhaustive Search Block Matching Algorithm with MPEG-4 Applications," in Proc. the 16th IEEE International Conference on Electronics, Circuits and Systems (ICECS'09), Hammamet, Tunisia, Dec. 13-16, 2009, pp. 787-790.

[4] Viitanen, Marko, et al. "Kvazaar: Open-Source HEVC/H. 265 Encoder." Proceedings of the 2016 ACM on Multimedia Conference. ACM, 2016. http://ultravideo.cs.tut.fi/#encoder -last accessed April 24th, 2017.

[5] https://www.xilinx.com/products/design-tools/vivado.html

[6] https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0041-zc7000-video-and-imaging-kit-hub.html.

[7] https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_3/SDK_Doc/concepts/sdk_c_bsp_internal.htm

[8] https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html