

FPGA-Based Hardware/Software Codesign for Video Encoder on IoT Edge Platforms

Cuong Pham-Quoc[1,2(✉)]

[1] Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet street, District 10, Ho Chi Minh City, Vietnam
[2] Vietnam National University - Ho Chi Minh City (VNU-HCM), Thu Duc, Ho Chi Minh City, Vietnam
cuongpham@hcmut.edu.vn

Abstract. Recently, image/video-based applications have been widely used for many domains, such as traffic, medical, or robotics. In this context, IoT-based systems with video processing implemented on edge devices have been applied extensively, for example, surveillance, object detection and monitoring, or intelligent home/building systems. Although algorithms for images/videos encoder with various standards produce good results, they usually require high computing power that is limited with edge platforms. Therefore, we propose an efficient hardware/software codesign approach to accelerate the video encoder process with FPGA-based IoT edge computing platforms in this work. The design flow exploits the effectiveness of the high-level synthesis process to quickly and efficiently generate FPGA-based hardware accelerator cores to improve system performance. We use the H.264 encoder as our case study to verify the proposed design flow and evaluate the accelerator ability compared to general-purpose processors. An FPGA Ultra96-v2 edge computing board is used for conducting experiments with a certified dataset. Experimental results show that we obtain speed-ups by up to 14.9× compared to a 4-core ARM processor. Our experimental system also saves up to 6.24× energy consumption compared to the ARM processor.

Keywords: FPGA · Hardware accelerator · H.264 encoder · Edge Computing · IoTs

1 Introduction

With the rapid development of technology and the internet, videos are used in various areas of the internet ranging from entertainment to supervising and monitoring. However, one of the most critical issues of network streaming video applications is the communication network bandwidth and the cost of transferring recorded videos from cameras to cloud storage or data centers. One promising approach to tackle this issue is to use edge computing platforms to process or pre-process videos and transfer only critical frames of objects detected

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14108, pp. 82–96, 2023.
https://doi.org/10.1007/978-3-031-37117-2_7

from videos to servers or cloud services. Edge computing platforms offer several advantages over traditional cloud computing solutions [3]. Edge computing reduces latency by processing data closer to where it's generated, reducing the time it takes for data to travel to and from the cloud. This ability is particularly beneficial for applications that require real-time processing, such as industrial automation, self-driving cars, and object detection.

Although this promising approach can be a solution for high-traffic and cost-ineffective video data transferring, low computing performance and limited energy capacity and storage of edge computing platforms are obstacles that should be considered carefully [15]. Therefore, in this work, we propose an approach for acceleration video encoders on FPGA-based edge computing platforms to overcome the obstacles of edge computing performance with energy efficiency. Our proposed approach first profiles the encoders for extracting the computationally-intensive functions that are good candidates for accelerating by FPGA fabrics. We then use high-level synthesis (HLS) tools to exploit different optimizations, such as loop unroll or pipeline, to generate the most optimized accelerator core. Finally, the system is built based on the hardware accelerator architecture [12] to exploit the advantages of both general-purpose processors and FPGA-based accelerator cores.

Due to its optimization, the H.264 video compression standard, also known as MPEG-4 Part 10, is one of the most well-known and widely used video coding standards for video streaming or video-based object monitoring. H.264 offers several advantages compared to other standards, such as high-quality video compression, enabling efficient video content storage and transmission without compromising image quality. Hence, we use the H.264 video encoder as our case study to validate the proposed approach. We develop our experimental system with the Ultra96-v2 FPGA edge computing platform to evaluate the acceleration ability compared to an embedded ARM processor. Experimental results with HD videos in Derf test media collection [23] show that our system achieves speed-ups by up to 14.9× compared to a 4-core ARM processor at 1.2 GHz.

The main contributions of our paper can be summarized in three folds.

1. We propose an efficient hardware/software codesign approach to accelerate the streaming video encoder for edge computing platforms on FPGA devices. A high-level synthesis process is used to generate hardware descriptions for the accelerator cores.
2. We present a case study with a well-known video coding format H.264 and the Ultra96-v2 FPGA board.
3. We show our experimental results with HD videos to reference other future studies.

The rest of the paper is organized as follows. Section 2 presents the background and related work. We introduce our proposed design approach in Sect. 3. Section 4 shows the implementation of the H.264 encoder as our case study. Experiments with HD videos on the Ultra96-v2 FPGA board are discussed in Sect. 5. Finally, Sect. 6 concludes our paper.

2 Background and Related Work

This section presents the background for video codec in general and H.264 in details. We then discussed related work in the literature focusing on acceleration video codec on FPGA.

2.1 Video Codec & H.264 Encoder

A video compression process requires two main components *encoder* and *decoder*. While the encoder encapsulates original video streams to compressed bitstreams with reduced data size for transferring and storing, the decoder converts the encapsulated bitstreams back to the original videos. The encoder usually includes three main functions for compressing: prediction model, spatial model, and entropy encoder. The decoder converts the compressed videos to original ones by using coefficients and prediction parameters of the encoder to calculate the entropy decode function, inverse transform, and reconstruction. Figure 1 summarizes the CODEC process based on the H.264 format [14].

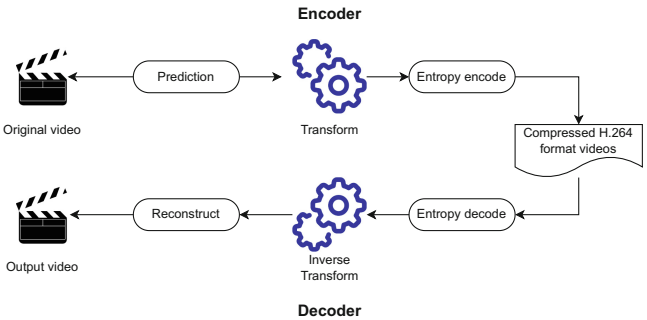

Fig. 1. The generic process of video encoder and decoder based on the H.264 format

As mentioned above, this work uses the H.264 encoder as a case study to demonstrate our proposed approach for FPGA edge computing platforms. The H.264 video compression standard uses hybrid video coding to encode and decode video data efficiently. The encoding process involves several steps, starting with video pre-processing, including resizing, color space conversion, and noise reduction. Figure 2 presents the syntax of the H.264 video compression standard. First, a sequence of an original video's frames is encoded to Network Adaption Layer Units (NALUs) packages. Each package stores parameters and a list of slices representing video frames. A slice consists of a header with encoding information followed by a list of macroblocks that encode a 16×16 pixels block of a video frame. Next, the encoding process applies compression techniques, such as intra-prediction, inter-prediction, and transform coding, to reduce redundancies in the video data and improve compression efficiency.

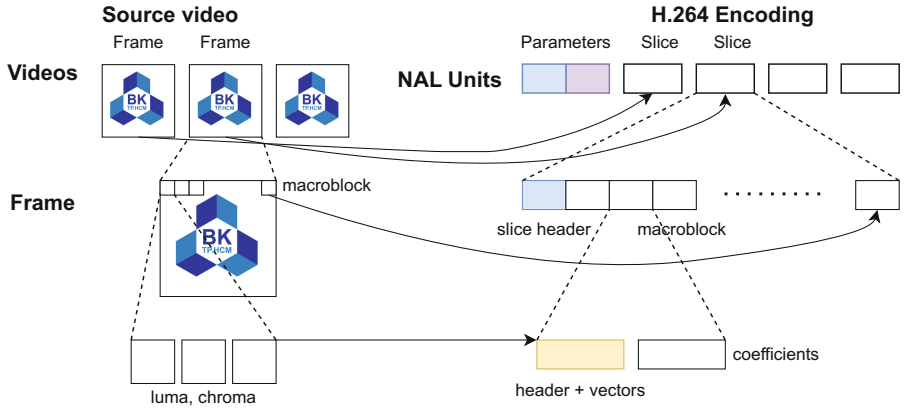

Fig. 2. The H.264 encoding process

2.2 Related Work

In recent years, many studies have focused on acceleration video encoders by hardware implementation. An entropy-based encoder for 8K video implemented on FPGA is introduced in [17]. In this work, the binarizer is customized, and pipeline techniques are exploited for massive parallelism to improve performance. However, this approach targets ultra-HD videos and is used in high-performance computing only. Researchers in [20] present their OpenCL-based and FPGA-powered H.266 acceleration. This approach implements the transformation and quantization steps on FPGA to improve performance. The experimental system is built with high-end FPGA devices instead of edge computing platforms like our work. Authors of [9] accelerate hexagon-diamond search (HEXDS) block matching algorithm on FPGA to improve performance of High-Efficiency Video Coding (HVEC) standard. This research only accelerates part of the encoding process and targets another compression standard. A heterogeneous CPU-FPGA computing architecture is used for HEVC video encoder [13]. The proposed system targets the OpenPOWER high-performance computing platform. A high-level synthesis approach, like our work, is used for acceleration of interpolation filter in HEVC video standard [16]. The proposed system targets high-end FPGA with a Virtex 6 device. Researchers in [10] design and build an ASIC accelerator core with the 90-nm TSMC technology for improving the performance of the HEVC video encoder. Due to the ASIC approach, the work achieves much improvement in performance. Versatile video coding (VVC) with 64×64 2D discrete sine/cosine transforms is proposed in [4]. This work focuses on the transformation step instead of the entire encoding process.

3 Proposed Hardware/Software Codesign Flow

This section proposes our design approach based on high-level synthesis (HLS) to accelerate video streaming encoder with FPGA edge computing platforms. Figure 3 illustrates the proposed hardware/software codesign flow for accelerating video streaming encoder based on FPGA and HLS. As shown in the figure, the flow consists of four main steps, starting from the high-level programming languages source code and ending with an FPGA-based edge computing implementation.

3.1 Profiling

The flow is started with the certified source code for an encoder method. The main purpose of this step is to get profiling data to make the most optimized hardware/software partition to improve system performance with hardware computation. The source code (usually in C/C++) is executed and profiled with a general-purpose processor to identify computationally intensive functions and data communication. Data communication is considered because transferring data across the computing cores also contributes much to the execution time of the entire system.

3.2 Hardware/Software Partition

Based on profiling data of the previous steps, the time-consuming functions are selected to build with dedicated hardware computing cores on the FPGA fabrics to take advantage of the reconfigurable technology (high parallelism, a massive amount of resources, and quickly adapting the design change). However, hardware fabrics cannot host all software functions due to the capability of resources and suitability of the functions. Therefore, this step needs to partition the source code into the software and hardware parts. The software part will continue processing with a general-purpose processor, while dedicated hardware cores will perform the hardware part. Due to dedications and hardware processing, the performance of these cores is higher than that of software-based processing.

This step also defines the interface for data communication between software functions and hardware cores. The target platforms should be taken into account for introducing the most optimized hardware/software interconnect. This interface can be a bus, crossbar, share memory, or network-on-chip based on the type of general-purpose processors that execute the software part. This software-hardware interconnect plays an important role in the system because communication overhead may require up to 50% execution times [11].

3.3 High-Level Synthesis

After determining functions accelerated by hardware-dedicated cores previously, this step generates HDL-based (hardware description language) processing cores.

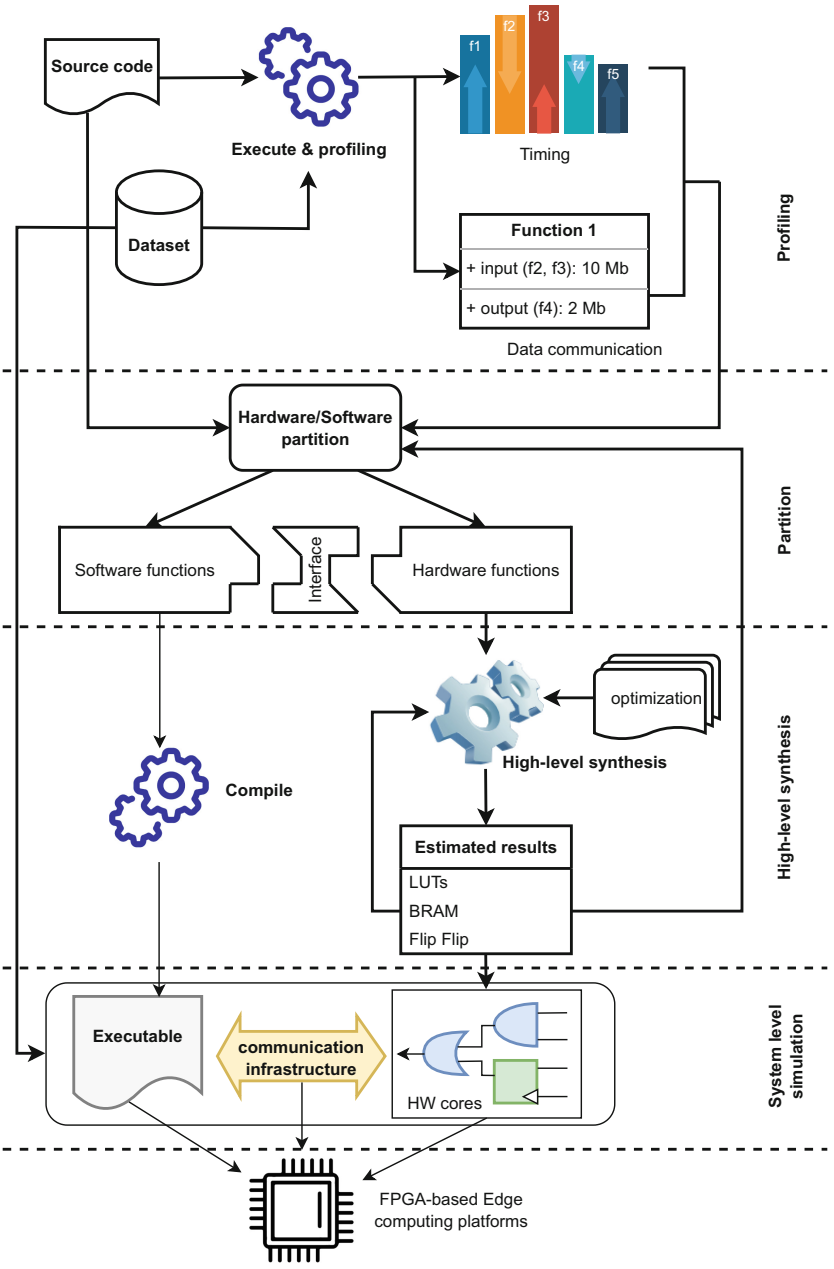

Fig. 3. The proposed design flow for video streaming encoder with HLS and targeting FPGA edge computing platforms

Although HDL-based cores can be developed manually by hardware designers, in this work, we use high-level synthesis (HLS) tools such as Vitis [22] from Xilinx or Stratus HLS [2] from Cadence for automatically creating the cores in HDL. HLS can offer many advantages compared to manual implementation, such as efficiently building and testing designs or exploiting various optimizations [8].

As shown in Fig. 3, this step takes optimization rules into account to generate HDL-based cores with estimated results in terms of the number of resources, working frequency, and latency. The HLS process can be performed in several iterations to find the most optimized solution. However, we sometimes need to redo the partition step since the IoT edge platforms' hardware resources may not be enough for the hardware cores.

3.4 System Level Simulation & Implementation

As the last step of the flow, after verifying with the high-level synthesis process, all parts are integrated and simulated during this step. Although the HLS process simulates and verifies the generated cores, simulation in this step is at the system level instead of the core level.

Depending on the target platforms, the general purpose processors can be embedded hardwired processors (for example, ARM or PowerPC) or softcore processors (for example, RISC-V or NIOS-II). These general-purpose processors are responsible for executing the software part of the application, while hardware-dedicated cores perform computationally intensive functions. This process needs to be supported by a system development kit provided by the edge target platforms such as Vivado from Xilinx as our experiments.

4 Case Study with H.264 Encoder

This section presents our case study with the H.264 encoder implemented on the Ultra96-v2 FPGA edge computing platform.

4.1 Source Code Analysis

H.264, as mentioned above, is a well-known video compression standard. Many certified H.264 encoder source codes exist, such as JM software [7] or ITU-T H.264 [6]. This work uses the x264 open-source software from VideoLAN Organization [19] as the reference source code. Figure 4 shows the main parts of the H.264 encoder implemented by VideoLAN software. The four processes, Filter, Analysis, Encode, and Entropy, are the four main computationally intensive functions.

The source code is then profiled with the perf tools [18] for extracting the timing values of functions. Although many profiling tools can be used for this purpose, we use perf because it is already integrated into most Linux kernels. Table 1 presents the perf results for the top computationally intensive source code functions.

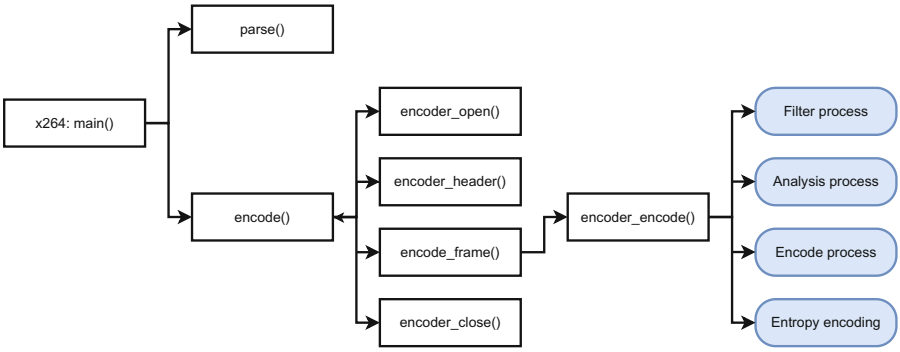

Fig. 4. Overview of H264 encoder main functions

Table 1. Timing profiling results with the perf tool for H.264 encoder

Function name	Overhead
x264_pixel_satd_8x4	26.12%
x264_pixel_avg_16x16	24.81%
x264_pixel_sad_16x16	14.67%
get_ref	14.24%
quant_4x4x4	10.43%
others	9,73%

The table shows that the `get_ref` function is also time-consuming along with the four functions `x264_pixel_satd_8x4`, `x264_pixel_avg_16x16`, `x264_pixel_sad_16x16`, and `quant_4x4x4`. However, this function collects data from external memory that an FPGA-based accelerator core cannot accelerate. Therefore, only the four functions, `x264_pixel_satd_8x4`, `x264_pixel_avg_16x16`, `x264_pixel_sad_16x16`, and `quant_4x4x4`, are suitable for accelerating by FPGA fabrics. Consequently, in this work, we optimize the five functions with HLS to generate accelerator cores for these functions. The following section presents the optimization techniques applied to the HLS processes.

4.2 High-Level Synthesis Optimization

High-level synthesis (HLS) [5] generates hardware circuits from high-level programming languages such as C or MATLAB. HLS has several advantages over traditional hardware design methods such as RTL (register transfer level). Compared to the traditional approach, HSL offers many advantages, such as higher productivity, faster design iterations, improved design quality, better portability, and reduced time to market. In this work, we exploit the faster design iterations of HLS by considering different optimization approaches for the four functions mentioned above. Those are *pipeline*, *unroll*, and *arry partition*. The *pipeline*

technique helps reduce the initiation interval of a loop or a function by applying concurrent operations. The *unroll* approach produces a copy of RTL logic for computing multiple operations concurrently instead of a single block executed multiple times. Finally, the *array partition* optimization creates HDL description for multiple small blocks of memory or registers instead of a large memory [22].

Table 2 shows the HLS estimation results for the above four functions regarding latency and hardware resource usage. This work uses the Xilinx Vitis tool [1] to conduct the high-level synthesis process. Although we consider various optimization techniques for each function, we only show the estimation results with the most optimized methods in this table. Furthermore, the table shows that the *pipeline* approach is used for all functions. Therefore, all four cores use many flip flops (FFs) as pipeline registers and Look-up tables (LUTs) for computing functions.

Table 2. HLS estimation results

Function	Optimized techniques	Resource estimates	Latency
x264_pixel_satd_8x4	pipeline	DSP: 1	11.370μs
	array partition	FF: 2165	
		LUT: 2340	
x264_pixel_avg_16x16	unroll	DSP: 96	75.360μs
	pipeline	FF: 3672	
		LUT: 6178	
x264_pixel_sad_16x16	array partition	DSP: 0	11.050μs
	pipeline	FF: 2075	
		LUT: 2364	
quant_4x4x4	array partition	DSP: 2	1.900μs
	pipeline	FF: 4088	
		LUT: 3791	

4.3 System Implementation

After optimizing and generating HDL-based hardware descriptions for the four accelerator cores, we build our testing system based on the hardware accelerator computing paradigm [12]. Our case study uses the Xilinx MPSoC Ultra96v2 edge computing board [21] as our evaluation platform. Our testing system features a 4-core ARM Cortex functioning as a host processor and FPGA fabrics, which can be utilized to construct different components such as local memory (buffer), exchange registers for handling IP cores, and accelerator IP cores for computing the time-consuming functions mentioned above. Figure 5 illustrates the platform's overall setup. All components inside the MPSoC FPGA device are connected through a standard AXI bus. Due to space constraints, only the system's primary features are displayed in the image.

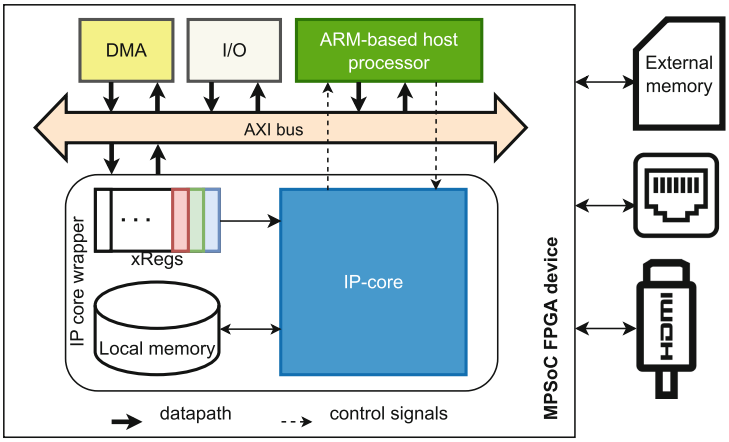

Fig. 5. Overview the FPGA-based edge computing system for H.264 encoder

4.4 Further Improvement with the Pipeline Technique

Due to the multi-processing cores implemented (hardware IP cores and a general-purpose processor), we can further improve the system's performance with the pipeline technique to achieve better performance. In general, while hardware accelerator cores calculate the accelerated functions, the general processor fetches new data to the local memory of the IP cores and processes the output of these cores. Figure 6 illustrates our scheduling for pipeline processing over seven cycles. Stages with the same color represent the processing of one piece of data. Our pipeline processing is divided into four stages. The general purpose processor executes one stage with the `get_ref` function at first and other functions when the hardware cores are finished following. The four hardware accelerator cores designed in the previous steps perform the rest three stages. Due to the pipeline approach, we need a more significant local memory for storing the current being processed data and the next piece of data.

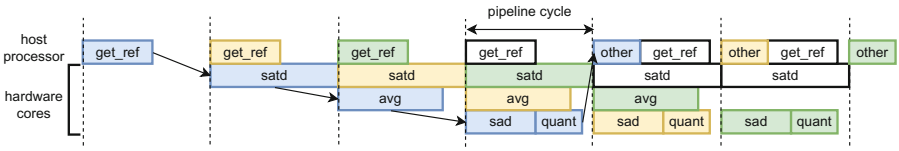

Fig. 6. Pipeline scheduling for the system

5 Experiments

In this section, we present our experiments to validate and estimate the acceleration ability of the above system. At first, the experimental setup is introduced. Then, we summarize the results with our experiments to illustrate the goals of our work.

5.1 Experimental Setup

The system described in Sect. 4 is evaluated using the Ultra96v2 board, which contains a Xilinx Zynq FPGA featuring over 70K Look-up Tables, 950 KB Block RAMs, and a 4-core ARM Cortex with a maximum clock speed of 1.5GHz. IP cores generated by HLS, Verilog-based exchange registers and local memory descriptions are utilized to construct all IP core wrappers within the FPGA's programmable logic part of the MPSoC device. An AXI bus connects the computational cores to the ARM Cortex as the host processor.

To assess the effectiveness of the proposed computing system, we compare our system with the processing capabilities of only the ARM processor. The following platforms are examined in detail:

1. *Our system* is a hardware accelerator architecture consisting of both the ARM processor and hardware cores for the four computationally-intensive of H.264 encoder, including `x264_pixel_satd_8x4`, `x264_pixel_avg_16x16`, `x264_pixel_sad_16x16`, and `quant_4x4x4`. The hardware cores perform the four functions when they are invoked, while the ARM processor handles the other parts of the encoder. In this setup, the ARM processor operates at its maximum frequency (1.5 GHz), while the hardware cores run at 230 MHz.
2. *The ARM system* uses all cores of the 4-core ARM processor for processing the entire H.264 encoder in parallel with the support of the x264 source code. Like our system, the ARM processor functions at the maximum frequency of 1.5 GHz for a fair comparison. However, the FPGA fabrics (hardware cores) are disable since only the ARM processor is utilized.

To test and compare our system with the ARM system, uncompressed videos from the Derf media collection are used [23]. In addition, we randomly chose three HD videos from the dataset to evaluate the proposed system's performance and energy efficiency. The following section presents our experimental results.

5.2 Experimental Results

At first, our system is synthesized with Xilinx tools to configure the Ultra96v2 board. The synthesis report shows that our system can function at 230 MHz and requires 4.208 W of power consumption. Figure 7 depicts the percentage of hardware resources our system used compared to the total amount of resources offered by the FPGA device. The figure shows that we use a lot of BRAM compared to the other types of resources due to our pipeline approach. More precisely,

many block RAMs are used for building local memory. However, according to the figure, more room exists for increasing the hardware cores or replicating existing accelerators to improve system performance.

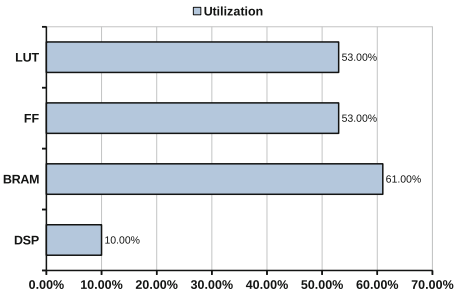

Fig. 7. Hardware resources usage for the whole system

Three HD videos from the dataset are collected to test the proposed system. Table 3 presents our performance in terms of execution evaluation of the system. According to the table, we achieve speed-ups by up to 14.9× compared to the ARM system. For the HD video *in_to_tree_420_720p50* (1280 × 720 resolution), we can process up to 47 frames per second.

Table 3. Performance comparison in terms of execution time (second)

Video input	#frames	Our system	ARM system	Output size
riverbed_1080p25	250	117.53	1057.78	11 MB
in_to_tree_420_720p50	500	10.53	156.88	648 KB
ducks_take_off_420_720p50	500	22.30	266.93	2.6 MB

Figure 8 compares the speed-ups of the proposed system with the ARM processor. As shown in the figure, we achieve speed-ups from 9× to 14.9×. The figure also depicts the energy consumption comparison between the two systems (energy = power × time). Compared to the ARM processor, our system is up to 6.24× energy efficiency.

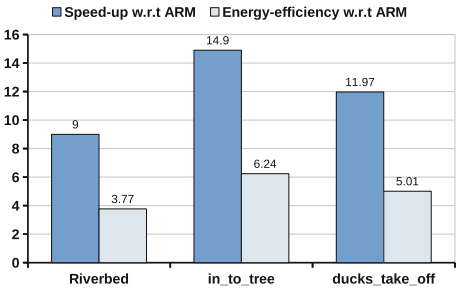

Fig. 8. Speed-ups and energy-efficiency with respect to the ARM system

6 Conclusion

The use of image and video-based applications has become widespread across various domains, including traffic management, medical fields, and robotics. With the rise of IoT-based systems, video processing has been implemented on edge devices for surveillance, object detection and monitoring, and intelligent home or building systems. However, while image and video encoder algorithms have produced satisfactory results, they typically require high computing power, which is constrained in edge platforms. To address this, we propose an efficient hardware/software co-design approach to accelerate the video encoder process using FPGA-based IoT edge computing platforms. Our design flow utilizes high-level synthesis to rapidly and efficiently generate FPGA-based hardware accelerator cores to enhance system performance. We use the H.264 encoder as a case study to verify our design flow and evaluate its accelerator capabilities compared to general-purpose processors. Our experiments with a certified dataset on an FPGA Ultra96-v2 edge computing board demonstrate that our system achieves up to 14.9× speed-up compared to a 4-core ARM processor, while also reducing energy consumption by up to 6.24× compared to the ARM processor.

Acknowledgement. We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU- HCM for supporting this study.

References

1. AMD Xilinx. Vitis™ unified software platform overview (2023). https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html. Accessed 01 Apr 2023
2. Cadence. Stratus high-level synthesis fastest path from specification to silicon (2023). https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-high-level-synthesis-ds.pdf. Accessed 01 Apr 2023

3. Caprolu, M., Di Pietro, R., Lombardi, F., Raponi, S.: Edge computing perspectives: Architectures, technologies, and open security issues. In: 2019 IEEE International Conference on Edge Computing (EDGE), pp. 116–123 (2019). https://doi.org/10.1109/EDGE.2019.00035
4. Garrido, M.J., Pescador, F., Chavarrías, M., Lobo, P.J., Sanz, C., Paz, P.: An FPGA-based architecture for the versatile video coding multiple transform selection core. IEEE Access **8**, 81887–81903 (2020). https://doi.org/10.1109/ACCESS.2020.2991299
5. Gupta, R., Brewer, F.: High-Level Synthesis: A Retrospective, pp. 13–28. Springer, Netherlands (2008). https://doi.org/10.1007/978-1-4020-8588-8_2
6. International Telecommunication Union. H.264.2: Reference software for itu-t h.264 advanced video coding (2016). https://www.itu.int/rec/T-REC-H.264.2. Accessed 01 Apr 2023
7. Joint Video Team. H.264/14496-10 AVC reference software manual (2009). https://iphome.hhi.de/. Accessed 01 Apr 2023
8. Martin, G., Smith, G.: High-level synthesis: Past, present, and future. IEEE Design Test Comput. **26**(4), 18–25 (2009). https://doi.org/10.1109/MDT.2009.83
9. Mukherjee, A.: VLSI architecture design of motion estimation block with hexagon-diamond search pattern for real-time video processing. In: 2021 IEEE 18th India Council International Conference (INDICON), pp. 1–6 (2021). https://doi.org/10.1109/INDICON52576.2021.9691531
10. Pastuszak, G.: Multisymbol architecture of the entropy coder for h.265/hevc video encoders. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems vol. 28, no. 12, pp. 2573–2583 (2020). https://doi.org/10.1109/TVLSI.2020.3016386
11. Pham-Quoc, C., Al-Ars, Z., Bertels, K.: A heuristic-based communication-aware hardware optimization approach in heterogeneous multicore systems. In: 2012 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2012 (2012). https://doi.org/10.1109/ReConFig.2012.6416720
12. Pham-Quoc, C., Heisswolf, J., Werner, S., Al-Ars, Z., Becker, J., Bertels, K.: Hybrid interconnect design for heterogeneous hardware accelerators. In: 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 843–846 (2013). https://doi.org/10.7873/DATE.2013.178
13. Qiu, Y., et al.: A heterogeneous HEVC video encoder system based on two-level CPU-FPGA computing architecture. In: 2021 IEEE 14th International Conference on ASIC (ASICON), pp. 1–4 (2021). https://doi.org/10.1109/ASICON52560.2021.9620382
14. Richardson, I.E.: The H.264 Advanced Video Compression Standard, 2nd edn. Wiley Publishing (2010)
15. Roman, R., Lopez, J., Mambo, M.: Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. Future Gen. Comput. Syst. **78**, 680–698 (2018). https://doi.org/10.1016/j.future.2016.11.009
16. Sjövall, P., Rasinen, M., Lemmetti, A., Vanne, J.: High-level synthesis implementation of an accurate hevc interpolation filter on an FPGA. In: 2021 IEEE Nordic Circuits and Systems Conference (NorCAS), pp. 1–7 (2021). https://doi.org/10.1109/NorCAS53631.2021.9599653
17. Tao, L., Gao, W.: A hardware implementation of entropy encoder for 8k video coding. In: 2022 IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6 (2022). https://doi.org/10.1109/ICME52920.2022.9859988
18. The Linux Kernel Archives: perf: Linux profiling with performance counters (2023). https://perf.wiki.kernel.org/. Accessed 01 Apr 2023

19. VideoLAN Organization: x264, the best h.264/avc encoder (2013). http://www.videolan.org/developers/x264.html. Accessed 01 Apr 2023
20. Waidyasooriya, H.M., et al.: Opencl-based design of an FPGA accelerator for h.266/vvc transform and quantization. In: 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1–4 (2022). https://doi.org/10.1109/MWSCAS54063.2022.9859281
21. Xilinx. Zynq ultrascale+ mpsoc (2021). https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html. Accessed 01 Apr 2023
22. Xilinx, A.: Vitis high-level synthesis user guide (ug1399) (2022). https://docs.xilinx.com/r/en-US/ug1399-vitis-hls. Accessed 01 Apr 2023
23. Xiph.org. Video test media [derf's collection] (2016). https://media.xiph.org/video/derf/. Accessed 01 Apr 2023