A Heterogeneous HEVC Video Encoder Based on OpenPOWER Acceleration Platform

Chenhao Gu, Yang Chen
IBM Corporation
State Key Laboratory of ASIC and System, Fudan University
Shanghai, China

Yanheng Lu, Pengfei Gou, Yong Lu, Yang Dai, Yue Xu, Yang Liu
IBM Corporation
Shanghai, China

Yibo Fan
fanyibo@fudan.edu.cn
State Key Laboratory of ASIC and System, Fudan University
Shanghai, China

ABSTRACT

This paper describes a heterogeneous HEVC video encoder system based on the OpenPOWER platform. Our design leverages the Coherent Accelerator Processor Interface (CAPI) on the OpenPOWER, which provides cache-coherent access for FPGA. This technology highly improves CPU-FPGA data communication bandwidth and programming efficiency. X265 is optimized on the OpenPOWER platform to improve its performance with both architecture specific methods and hardware-acceleration methods. For hardware acceleration, frame-level acceleration and functional-unit-level acceleration are introduced and evaluated in this work.

KEYWORDS

HEVC, x265, CAPI, OpenPOWER, heterogeneous computation

1 INTRODUCTION

High Efficiency Video Coding (HEVC) standard is now one of the most widespread video coding standards. Compared with H.264/AVC, HEVC achieves about twice the compression efficiency [1].

X265 is an open-source encoder project which aims to deliver the world's fastest and most efficient HEVC video encoder. Although x265 has been developed efficiently with many optimization techniques, it is still not able to support 8K UHD real-time encoding even at ultrafast setting. Therefore a heterogeneous HEVC video encoder that involving FPGA accelerator is proposed in this paper.

There are three big challenges for the implementation of a heterogeneous HEVC video encoder. Firstly, X265 is run in multi-thread mode, which means the communication between hardware and software is a big challenge. Secondly, latency is very important as each FPGA engine will be called many times. Thirdly, to encode high-resolution (HD) videos, bandwidth is a big problem as both original pixels and reference pixels together with some intermediate data are transmitted between host memory and FPGA.

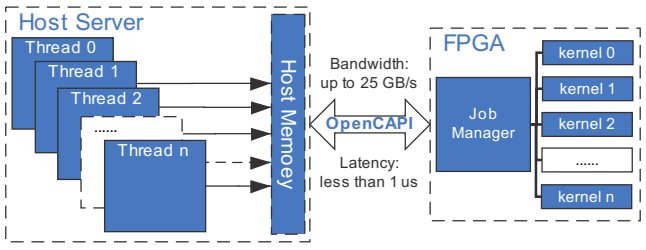

Figure 1: Multi-thread and multi-kernel architecture

Therefore, a multi-thread and multi-kernel architecture based on CAPI is proposed in this work, as shown in Fig. 1. The software part is run in multi-thread mode and several hardware kernels are deployed on FPGA. As CAPI is a cache-coherent interface, hardware kernels can directly access host memory to reduce the latency. Also, the bandwidth of OpenCAPI is up to 25 GB/s [2], which is one of the highest interfaces now.

The rest of this paper is organized as follows. Software optimization on POWER is introduced in Section 2. Both frame-level and functional-unit -level acceleration are introduced in Section 3. Section 4 concludes this work.

2 SOFTWARE OPTIMIZATION

Our optimization for performance is based on the system level. IBM POWER9 processor has some advantages on the number of SMT (Simultaneous Multi-Threading) of each core and some built-in vector instructions which are helpful to improve x265 performance.

2.1 Binding

The best application performance can be usually obtained by keeping its parallel threads as close to the memory as possible. Although system mechanisms like Linux thread scheduler would do this automatically, the reality is that most HPC applications benefit greatly from manually placing threads on different processor cores. The binding can be useful for CPU-intensive programs that experience few interrupts. Our server contains 2 nodes, each of which contains 22 cores, and each core contains 4 threads. By using Linux binding command while running the x265, we can force the multi-thread program to be dispatched to the certain cores to ensure each core execute two threads at one time – a single POWER9 core can handle 2 vector 128-bit operations every cycle [3], as shown in Fig. 2.

2.2 POWER9 Vector Instruction

Replacing some functions in x265 with some vector instructions provided by POWER9 can improve the encoding speed. For example, it can be found that the original x265 uses 3 instructions to calculate the vector absolute differences. It can be optimized by using one single vector instruction called vec_absd on the POWER9 server. The functionality of the three functions mentioned above are included in vec_absd, which computes the absolute differences of the vector elements in two vector arguments and places the absolute differences into the result

$$vec\_absd(vec1, vec2) = vec\_sub(vec\_max(vec1, vec2), vec\_min(vec1, vec2)). \quad (1)$$

Compared with other platforms, an average performance improvement of 52.5% is achieved on the POWER platform by binding and function replacement, as shown in table 1.

Table 1: Results of Software Optimization

Sequence	x86[1]	P9[2]	vabsd[3]	4tpc[4]	1tpc	2tpc
Kinomo	41.0	47.9	50.0	47.8	63.0	61.6
ParkScene	42.5	46.2	46.3	44.2	60.2	61.5
Cactus	39.5	47.7	48.8	48.0	57.9	60.5
BasketballDrive	41.0	46.0	47.3	48.4	64.1	63.2
BQTerrace	37.7	49.6	50.1	50.5	58.1	60.7
Traffic	26.0	33.0	34.3	33.6	36.1	37.8
PeopleOnStreet	17.2	25.1	25.2	24.9	25.0	27.3

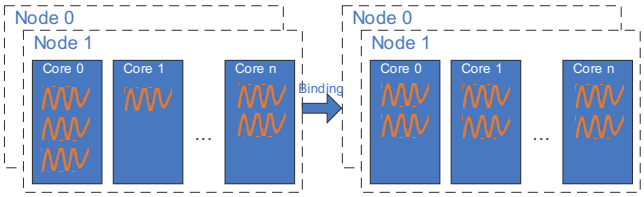

Figure 2: Block Diagram of Binding

3 HARDWARE ACCELERATION

For hardware acceleration, some parts of the x265 are replaced with hardware units and can be divided into two categories, frame-level acceleration, and function-unit-level acceleration.

3.1 Frame-level Acceleration

For frame-level acceleration, the intra prediction module of x265 is kept and performed on POWER while the whole inter prediction module of x265 is implemented as a hardware unit and deployed on the FPGA. In this work, one frame is compression using intra prediction, followed by one frame compressed with inter prediction.

The hardware implementation is deployed on the FPGA at 200 MHz, which supports up to 1080p@60fps. Table. 2 shows the resource utilization of the proposed hardware engine.

Table 2: Resource Utilization of Inter-frame Encoder

Site Type[5]	Used	Available	Util%
LUTs	410794	1182240	34.75
Registers	192161	2364480	8.13
Block RAMs	2096	2160	97.04
DSPs	587	6840	8.58

[1]Intel(R) Xeon(R) Gold 6148 CPU@3.70GHz 40 cores 80 threads
[2]IBM POWER9 CPU@3.80GHz 44 cores 176 threads
[3]Vector absolute difference
[4]Thread per core
[5]Xilinx xcvu9pflgb-2104-2L

Besides, some algorithms of the inter-frame coding in x265, such as rate distortion optimization and rate control, are not suitable for hardware implementation. The proposed hardware design simplifies these parts, which inevitably leads to the BD-rate increase. The proposed frame-level acceleration scheme reaches about 1080p@120fps encoding with an average BD-rate increase of about 15.0% compared with x265.

3.2 Functional-unit-level Acceleration

The frame-level acceleration has disadvantages on the BD-rate loss. The reason lies in the simplification of the inter-frame coding for hardware design. For functional-unit-level acceleration, some sub-functions of x265 are implemented as hardware engines. These functions are computationally expensive and suitable for hardware implementation. The small latency of OpenCAPI is significant for functional-unit-level acceleration since these sub-functions are called thousands of times per second for HD video coding.

According to Márquez's work [4], motion estimation (ME) is one of the most time-consuming functions. The integer motion estimation (IME) function of x265 is implemented as a hardware engine and several engines are deployed on the FPGA. The required bandwidth request of the IME function is about 5.97 GB/s for 8K@30fps encoding, which can be satisfied by the high-bandwidth OpenCAPI.

Two modes of functional-unit-level acceleration are proposed in this work, embedded mode, and ahead mode. The difference between these two modes is the call time of the hardware engine.

For embedded mode, the x265 software sends a starting signal to the hardware engine and fetches results back after the hardware engine finishes the computation. However, the embedded mode is not suitable for sub-functions with huge data interaction. The data interaction between the hardware engine and x265 software is time-consuming even using the high-bandwidth CAPI.

For ahead mode, the computation operation of the hardware engine is ahead of corresponding x265 software. X265 can fetch the results from the host memory directly with negligible delay since the required results have been calculated during the computation operation of the hardware engine. However, only the functions at the first stage of the compression are suitable for ahead mode. Since IME is the first stage of the inter prediction, the IME engine adopts ahead mode in this work. The average coding speed increases by 10.4% by merely replacing IME with our hardware engine.

4 CONCLUSION

In this work, a heterogeneous HEVC video encoder is proposed. The open-source x265 is optimized with OpenPOWER architecture-specific methods. Two hardware acceleration methods, frame-level acceleration, and function-unit-level acceleration are also put forward. Frame-level acceleration features a high speedup rate in the sacrifice of image quality. For function-unit-level acceleration, two modes are proposed with negligible quality loss. Some hardware-friendly modules, such as IME, are implemented as hardware engines. The average coding speed increases by 10.4% by merely replacing the IME function with our hardware engine. For future work, more modules of x265 will be evaluated and integrated into the OpenPOWER platform using embedded mode or ahead mode to improve the image quality and coding speed.

REFERENCES

[1] J. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand. Comparison of the coding efficiency of video coding standards-including high efficiency video coding (hevc). *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1669–1694, Dec 2012.

[2] J. Stuecheli, W. J. Starke, J. D. Irish, L. B. Arimilli, D. Dreps, B. Blaner, C. Wollbrink, and B. Allison. Ibm power9 opens up a new era of acceleration enablement: Opencapi. *IBM Journal of Research and Development*, 62(4/5):8:1–8:8, July-Sept 2018.

[3] S. K. Sadasivam, B. W. Thompto, R. Kalla, and W. J. Starke. Ibm power9 processor architecture. *IEEE Micro*, 37(2):40–51, Mar 2017.

[4] G. Cebrián-Márquez, J. L. Martínez, and P. Cuenca. A pre-analysis algorithm for fast motion estimation in hevc. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 2013–2017, Phoenix, 2016. IEEE.