A Heterogeneous HEVC Video Encoder Based on OpenPOWER Acceleration Platform

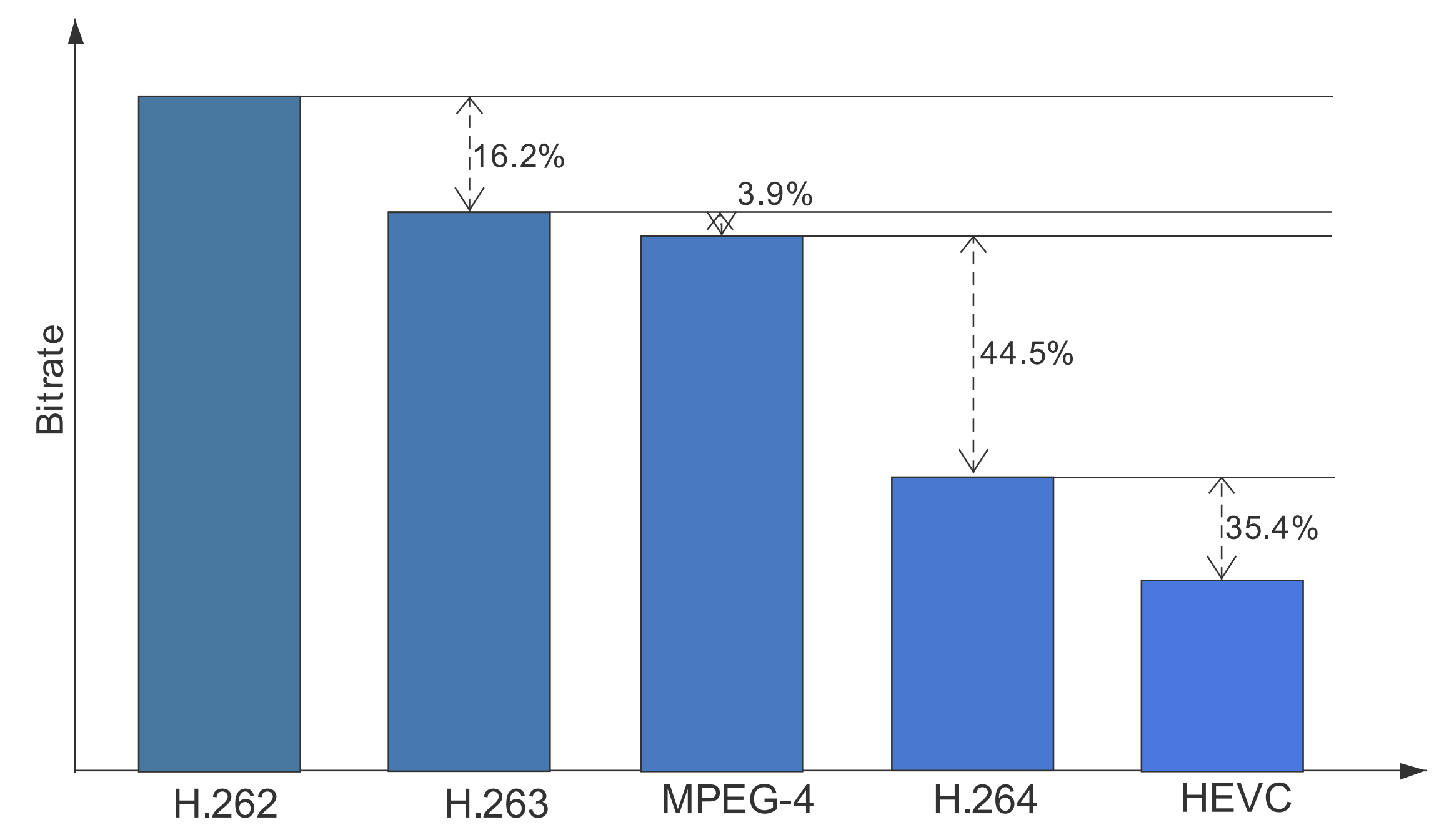
Chenhao Gu, Yang Chen, Yanheng Lu, Pengfei Gou, Yong Lu, Yang Dai, Yue Xu, Yang Liu and Yibo Fan

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.

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].

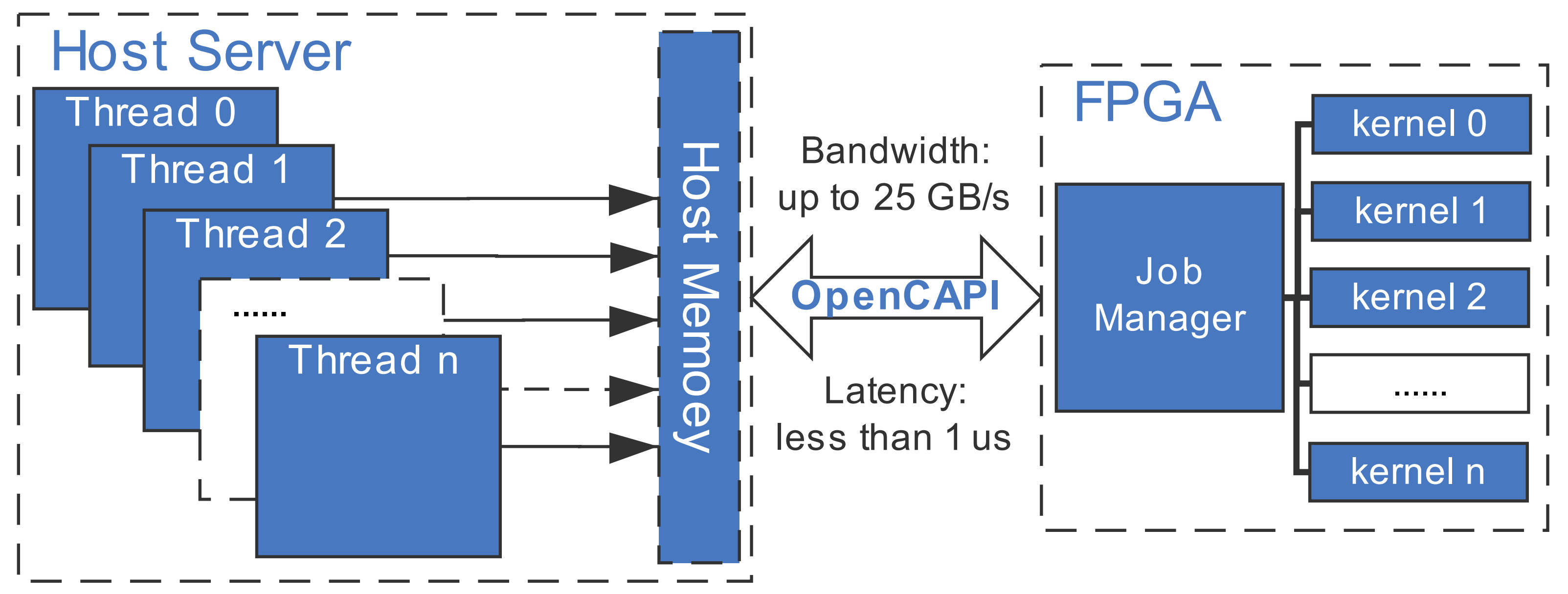


Comparison of video coding standards

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 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.

In this work, a multi-thread and multi-kernel architecture based on CAPI is proposed to solve the above problems. CAPI is a high-bandwidth, low-latency and software friendly interface [2] , which is suitable for video coding application.



Multi-thread and multi-kernel architecture

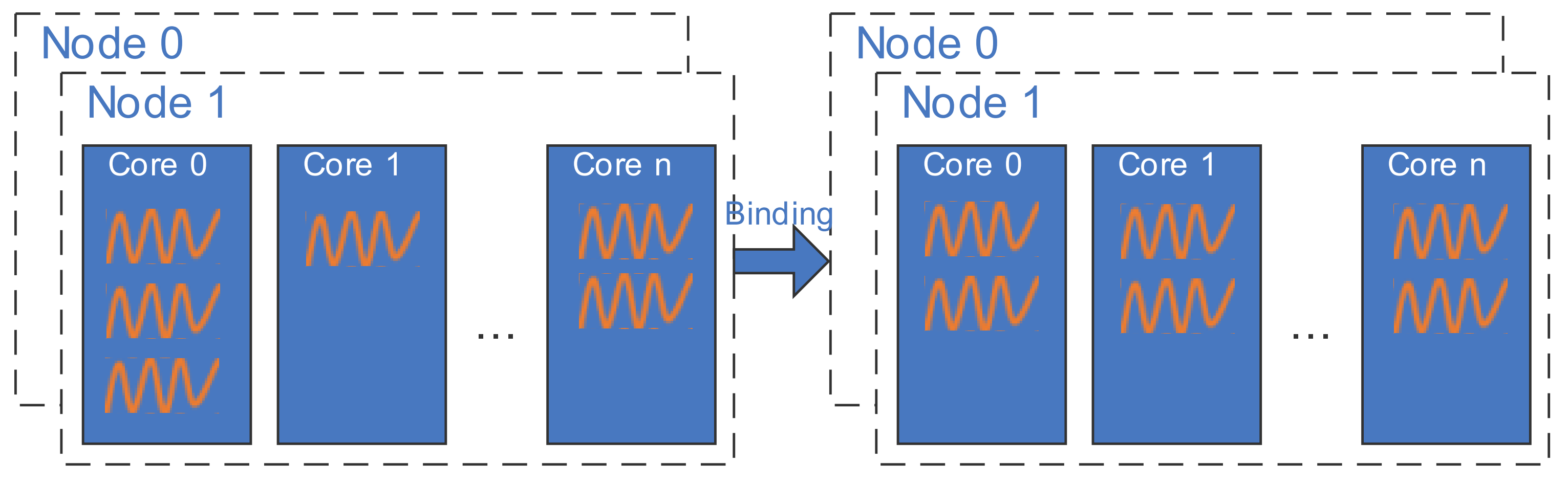
Software Optimization

Our optimization for performance is based on the system level. Different server platforms provide different functional strength. 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.

Most HPC applications benefit greatly from manually placing threads on different processor cores. By using Linux **binding** command (i.e. numactl) while running the x265, we can force the multi-thread program to be dispatched to the limited cores so that ensure each core execute two threads at one time – a single POWER9 SMT4 core can handle 2 vector 128-bit operations every cycle [3].

Replacing some functions in x265 with some **vector instructions** provided by POWER9 can also improve the coding speed. For example, it can be found that the original x265 uses three instructions to calculate the vector absolute differences. It can be optimized by using one single vector instruction called vec_absd() on the IBM POWER9 server.

$$vec\_absd(vec1, vec2) = vec\_sub(vec\_max(vec1, vec2)\,, \mathrm{vec\_min}(vec1, vec2))$$



Block diagram of binding

The function replacement on the IBM POWER9 server makes an average performance improvement of 2.2%, and binding makes another 21.8% average improvement. The total software optimization makes 52.5% average improvement compared with performance tested on x86 servers, and 24.5% average improvement compared with the original x265 tested on POWER9 servers. The encoding performance represented by the number of frames encoded per second(fps), is listed below respectively.

Encoding performance comparison between different machines with/without P9 vector instructions

Sequence	Resolution	x86[a]	P9[b] original	With vabsd[c]
Kimono	1920x1080	41.03	47.89	50.02
ParkScene	1920x1080	42.45	46.17	46.29
Cactus	1920x1080	39.49	47.73	48.84
BasketballDrive	1920x1080	40.97	46.02	47.3
BQTerrace	1920x1080	37.73	49.61	50.08
Traffic	2560x1600	25.99	33.01	34.32
PeopleOnStreet	2560x1600	17.17	25.09	25.21

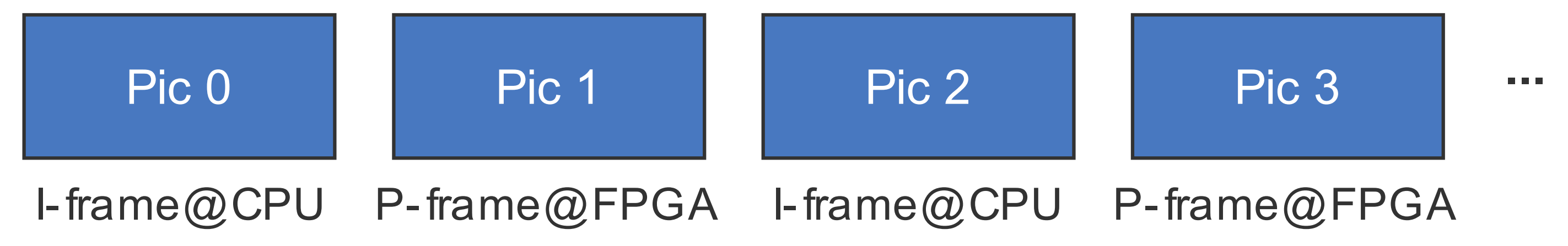
[a] Intel(R) Xeon(R) Gold 6148 CPU@3.70GHz 40 cores 80 threads
[b] IBM POWER9 CPU@3.80GHz 44 cores 176 threads
[c] Vector absolute difference

Encoding performance comparison between different binding strategies vs no-binding version

Sequence	Four threads per core		One thread per core		Two threads per core	
Kimono	47.78	↓4.48%	63.01	↑25.97%	61.59	↑23.13%
ParkScene	44.24	↓4.43%	60.21	↑30.07%	61.45	↑32.75%
Cactus	47.97	↓1.78%	57.85	↑18.45%	60.48	↑23.83%
BasketballDrive	48.43	↑2.39%	64.06	↑35.43%	63.16	↑33.53%
BQTerrace	50.47	↑0.78%	58.14	↑16.09%	60.71	↑21.23%
Traffic	33.59	↓2.13%	36.07	↑5.10%	37.79	↑10.11%
PeopleOnStreet	24.89	↓1.27%	25.00	↑0.83%	27.26	↑8.13%

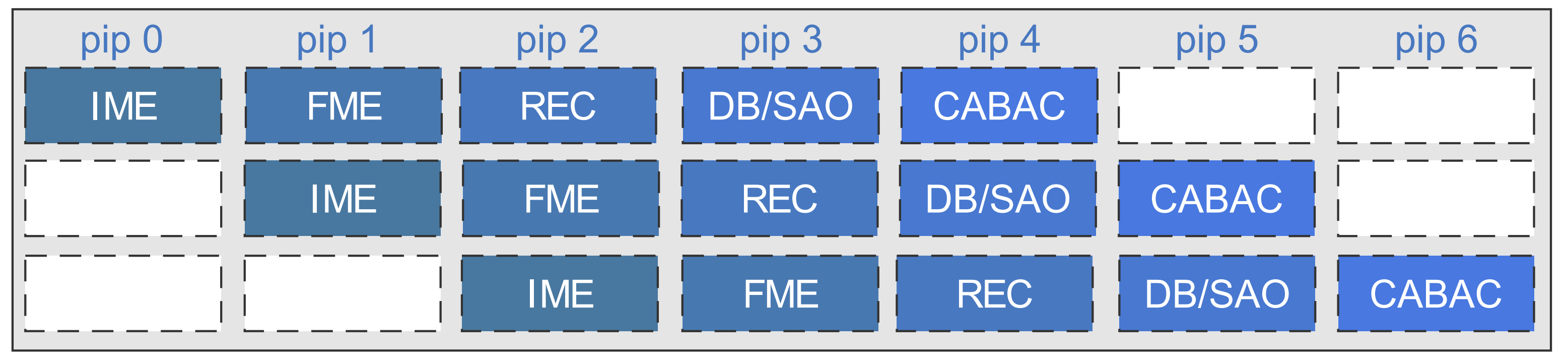
Hardware Acceleration

For **frame-level acceleration**, the intra prediction module of x265 is kept and performed on POWER9 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.



Frame types of frame-level Acceleration

The pipeline stages of the proposed hardware design are shown below. The hardware inter-frame encoder is divided into five pipeline stages, integer motion estimation (IME), fractional motion estimation (FME), reconstruction loop (REC), deblocking filter (DB) and context-adaptive binary arithmetic coding (CABAC).



Pipeline stages of proposed engine

The hardware implementation is deployed on the FPGA at **200 MHz**, which supports up to 1080p@60fps.

Resource utilization of inter prediction engine

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

[a] Xilinx xcvu9pflgb2104-2L

The frame-level acceleration has disadvantages on the BD-rate loss. BD-rate indicates the bit rate difference with the same image quality. The reason lies in the simplification of some parts of the inter prediction for hardware design. The proposed frame-level acceleration scheme reaches about **1080p@120fps** encoding with an average BD-rate increase of about 15.0% compared with x265.

For **functional-unit-level acceleration**, some sub-functions of the x265 inter prediction 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. Profiling results show that the ME consumes more than 50% of the total execution time. 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.

Resource utilization of IME engine

Site Type	Used	Available	Utils %
LUTs	47443	1182240	4.01
Registers	25896	2364480	1.10
Block RAMs	285	2160	13.19
DSPs	0	6840	0

[a] Xilinx xcvu9pflgb2104-2L

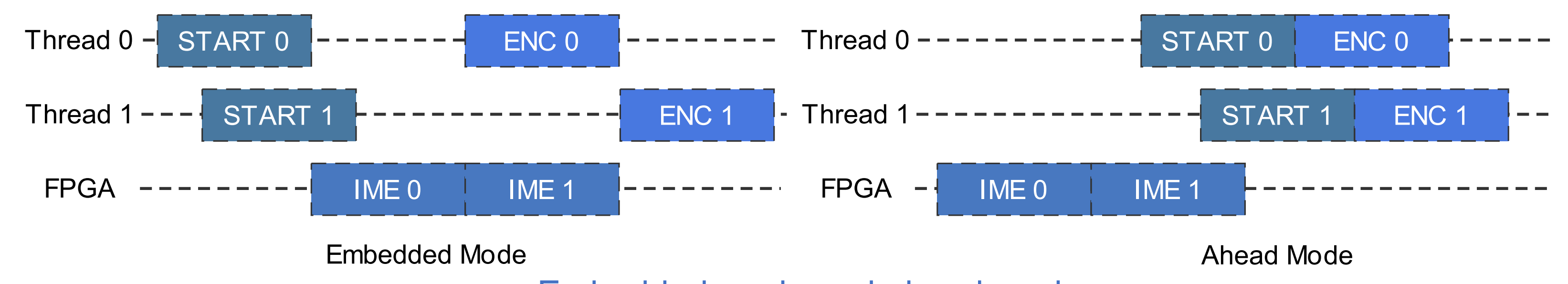
Two modes of functional-unit-level acceleration are proposed in this work.

- **Embedded mode**

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.

- **Ahead mode**

For ahead mode, the computation operation of the hardware engine is ahead of corresponding x265 software. The x265 can fetch the results from the host memory directly with negligible latency 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 coding speed increases by **10.4%** on average by merely replacing the IME function with our hardware engine.



Embedded mode and ahead mode

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 for image quality improvement and coding speed expedited. 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.

Reference

[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 et al., 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.