WHITE PAPER

FPGA Inline Acceleration

FPGA Inline Acceleration for Streaming Analytics

Authors

Meha Kainth
Intel® Corporation
Programmable Solutions Group
San Jose, USA
meha.kainth@intel.com

Dan Pritsker
Intel Corporation
Programmable Solutions Group
San Diego, USA
dan.pritsker@intel.com

Hong Shan Neoh
Intel Corporation
Programmable Solutions Group
San Jose, USA
hong.shan.neoh@intel.com

Abstract

This paper presents a method to implement FPGA inline acceleration for streaming analytics. The accelerator that is implemented on the FPGA fabric using the OpenCL™ approach with streaming pipes, processes data packets directly from the network via a 10 Gbps Ethernet (10GbE) interface and applies inline processing on the streaming data to monitor the signature waveforms in real time. The FPGA inline accelerator that acts as a co-processor to the CPU in streaming analytics platforms provides a scalable solution that can handle data as it grows in volume, velocity, and complexity.

The FPGA inline accelerator enables a hardware parallel platform that can handle analytics workloads of real-time data very efficiently. The capability to ingest data while performing inline processing for data conditioning can provide real-time insights from the streaming data. This approach offers a solution to extract information that resides in the data stream to generate rapid real-time decisions before the data becomes stale. In addition, the FPGA inline accelerator has multiple key advantages including low-latency interface, data locality, and modularity in input and deterministic response regardless of data rate or data format.

This paper provides the benchmark results for comparing the performance of inline acceleration that is implemented in an FPGA versus a system without a hardware accelerator in terms of latency and sustainable data rate. Results from the benchmark show that the FPGA inline accelerator has 22X lower end-to-end latency while maintaining data rate of 9 Gbps without dropped packets.†

Introduction

Streaming analytics is the ability to continuously analyze image processing and real-time data to extract meaningful information from it on the fly. The analysis can be in the form of mathematical calculations, statistical packet inspection [1]. Streaming analytics connects to various external data sources and sends certain data to downstream applications. This enables applications to integrate certain data into the application flow, or to update an external database with processed information [2]. The data can originate from the Internet of Things (IoT), mobile phones, and mobile devices, such as iPads, market data, social media, sensors, Web clickstream, and financial transactions [3]. Streaming analytics enables analysis of data as soon as it becomes available allowing the ability to analyze risks before they occur.

Table of Contents

The data in streaming analytics environment is processed before being stored in a storage database as opposed to traditional data analytics technologies that use batch processing techniques by storing data for a certain period before doing the analysis. This technology supports much faster decision making than possible. Furthermore, conventional approaches to streaming analytics involve downstream software tools running on CPU, that inspect and analyze data before forwarding it further downstream to systems and applications for consumption by end users. However, the ability for these tools to perform real-time analysis and generate alerts is limited by the performance of today's solutions used to Extract, Transform, and Load (ETL) data into downstream systems due to the latency they add between data collection and data analysis [4].

Mitigating these issues requires placing the beginning of the analytic pipeline as close to the point of ingress as possible and leveraging hardware acceleration for initial data analysis [4]. In this paper, we focus on our efforts to design and implement an FPGA-based hardware accelerator at the point of ingress such that it can process data at inline rate. The FPGA-based accelerator uses an Intel Arria® 10 FPGA and the Intel FPGA SDK for OpenCL with a Host Pipes application programming interface (API) to stream data into a kernel directly from a streaming I/O interface such as 10GbE. Our application applies transformation to the streaming data to monitor the signature waveforms in real time.

The paper first discusses the advantages of using FPGAs as a hardware accelerator for streaming analytics applications. Intel FPGAs support the OpenCL framework, which is an effective tool flow for heterogeneous computing. This framework supports OpenCL pipe semantics and 'Host Pipes' which is an efficient streaming interface integrated in the Intel FPGA OpenCL flow for low-latency streaming applications. Section III presents the related work. Section IV discusses the key advantages and data flow for using FPGA for inline acceleration. We discuss our design implementation and methodology in Section V. Section VI provides results for improvements in latency and sustainable data rates with FPGA as compared to a setup without a hardware accelerator. Section VII summarizes the paper.

Streaming Analytics Using FPGAs

FPGAs have high throughput and higher performance per watt efficiency as compared to general-purpose processors. Due to this, FPGAs have become an attractive and effective means of accelerating high-performance computing and data-centric applications as well as handling streaming analytics applications [5-8]. Some of these applications include information filtering [5] and social media, packet processing in network routers and firewalls [6], network intrusion prevention [9] and threat detection systems, video compression [10], low-latency market data feed arbitration for financial trading [11], photonic device simulation for scientific computation [12], and database analytics [13].

Streaming analytics require high-performance and low-latency processing of data streams. General purpose CPU may not be sufficient for real-time analytics. Intel FPGAs accelerate and aid in compute and connectivity required to collect and process the massive quantities of information by controlling the datapath. In addition to FPGAs being used as an accelerator, they can also directly receive data and process it inline before going through the CPU host system. This frees the processor to manage other system events and provide higher real-time system performance [14]. FPGAs' flexibility enables them to deliver deterministic low latency and high bandwidth. Some of the other factors for suitability of FPGAs for streaming analytics are:

- Lower latency: FPGAs can be connected closer to the streaming media, which means complete data processed with no extra transfer and load, eliminating the need for flow control
- Data locality: The processing element is close to the abundant internal memory bandwidth of an FPGA
- Energy efficiency
- Flexibility in handling various data rates and granularities due to built-in I/O interface and protocols

Intel FPGAs provide state-of-the-art solutions to enable designers to use FPGAs for hardware acceleration of streaming applications. The core components to enable this are a streaming interface and an OpenCL tool flow for FPGA.

High-Level Design Using OpenCL

FPGAs were traditionally programmed using hardware description languages (HDL) that are synthesizable, such as Verilog and VHDL. These languages include complex constructs for describing parallel simulations and timing delays that requires specialized skills. Recent improvements in tool flows have enabled development of OpenCL heterogeneous parallel programing framework for Intel FPGAs. The OpenCL standard naturally matches the highly parallel nature of FPGAs [15]. OpenCL allows the programmer to explicitly specify and control the thread-level parallelism, while developing in C-like programming language. This enables good match for FPGA development as it offers very high level of parallelism.

Unlike CPUs and graphics processing units (GPUs), where parallel threads can be executed on different cores, FPGAs offer a different approach. Kernel functions can be transformed into dedicated and deeply pipelined hardware circuits that are inherently multithreaded using the concept of pipeline parallelism. Each of these pipelines can be replicated many times to provide even more parallelism than is possible with a single pipeline. The Intel FPGA SDK for OpenCL compiler translates an OpenCL kernel to hardware by creating a circuit that implements each operation. These circuits are wired together to mimic the flow of data in the kernel. A CPU host program has access to standard OpenCL APIs that allow data to be transferred to the FPGA, invoking the kernel on the FPGA and returning the resulting data. This function to be accelerated is referred to as an OpenCL kernel. The FPGA's reconfigurability allows loading and unloading of different dedicated acceleration kernels that were designed for a particular type of workload [15].

Host Pipes

The OpenCL programming model allows software developers to easily tap into the computational power of hardware accelerator devices. To maximize throughput, in OpenCL, all data must be completely written to either the host or the device before it can be accessed. This sacrifices the latency [15] [16]. In OpenCL, the primary way to transfer data between host and accelerator has been via global memory. This results in an inherent trade-off between throughput and latency. One of the solutions is 'Streaming Pipes', which enable software developers to make use of OpenCL accelerators for low-latency streaming applications. To mitigate this issue, Intel has developed a streaming interface, called 'Host Pipes', for Intel FPGA SDK for OpenCL supporting Intel FPGAs. This interface is a vendor extension to the existing Intel FPGA OpenCL flow.

Host Pipe is a direct streaming interface from host CPU to OpenCL kernels that eliminates the latency overhead of waiting for data transfer through external global memory to complete before execution can start on the head of that data [16]. There are three components in this system—FPGA-side FIFO buffer created using on-chip memory, software-side FIFO buffer pinned in host memory, and direct memory access (DMA). Figure 1 shows the FIFOs in the host memory and the FPGA that continuously stream data from the CPU and OpenCL kernel respectively. The DMA transfers data between the two FIFOs in blocks to utilize full PCI Express* (PCIe*) bandwidth.

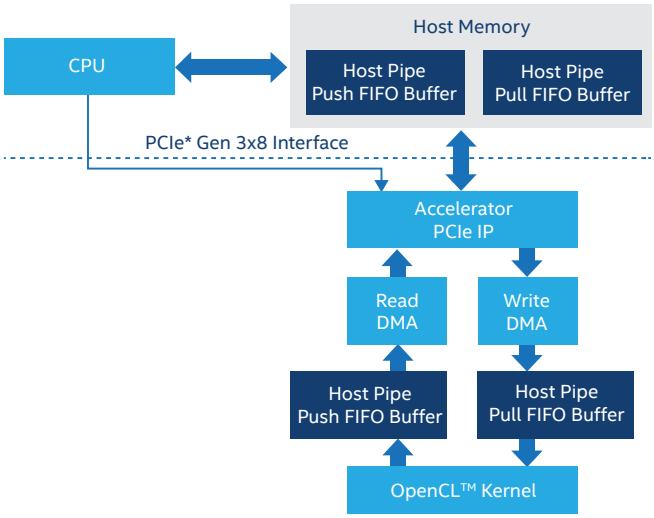

Figure 1. Host Pipe Prototype Architecture

Related Work

One of the main issues with FPGAs is the complexity of programmability and design flow in low-level HDL. Intel has developed the Intel FPGA SDK for OpenCL for much better programmability and to bring the benefits of FPGA hardware platforms to software programmers. Intel FPGA SDK for OpenCL targets OpenCL at Intel FPGAs to make application development easier. The advantage of working with OpenCL is that the same code can be easily targeted to different platforms, such as FPGA, GPU, and CPU for performance comparisons. Although the implementation still needs to be optimized and adapted to each platform to gain optimal performance, the evaluation process is much simpler [6].

Many research studies have been conducted in evaluating and comparing performance of several parallel and compute-intensive benchmarks in OpenCL targeting FPGA, GPU, and CPU. In [17], the authors optimized and ported a subset of the Rodinia benchmark suite to a Stratix® V FPGA using Intel FPGA SDK for OpenCL, and they compared the performance and energy efficiency between an Intel E5-2670 CPU and NVIDIA* K20c GPU. Their evaluation showed that in most benchmarks, only the energy efficiency was superior to the GPU, whereas both the performance and energy efficiency were better than those of the CPU. In [6], the authors use the OpenDwarfs benchmark suite, a suite of architecture-agnostic OpenCL kernels that capture common computation and communication patterns across a wide spectrum of scientific and engineering applications to study and compare the performance of OpenCL programming model on FPGAs. Certain optimization techniques were applied on GPU-based kernel for targeting Stratix V FPGAs. It is shown that FPGAs can result in a highly efficient pipeline achieving 91% of theoretical throughput for the structured grids dwarf.†

Pipe execution semantic is a feature in OpenCL 2.0 supporting concurrent kernel execution for improving throughput and latency. Due to their reconfigurability, FPGAs are well suited for OpenCL pipe execution because they allow multiple kernels to be executed in parallel over changing streams of data for streaming applications [8]. The authors in [8] have used the OpenCL pipe on Intel FPGA SDK for OpenCL for accelerating 2-dimensional (2D) vision algorithms to benefit pipe-based execution. They observed a 2.8X increase in throughput for tuned pipelined kernels, as compared to sequential execution.†

As an extension to the OpenCL pipe feature, a new feature called Host Pipe has been added to the Intel FPGA SDK for OpenCL to transfer data directly between host CPU to OpenCL kernels on FPGA [16]. The authors have developed a prototype with the Intel Arria 10 FPGA Development Kit. Results show high throughput, achieving up to 75% of PCIe Gen3x8 bandwidth for loopback kernel and up to 40X reduction in latency for an OpenCL Gzip application while maintaining 3 Gbps compression.†

In this work, we are using the Host Pipe interface for OpenCL kernel development on a Nallatech 385A acceleration card with an Intel Arria 10 FPGA. Using this low-latency interface, we have developed an inline FPGA accelerator with a CPU and benchmarked our results against a stand-alone CPU for streaming analytics applications.

Inline FPGA Acceleration

Due to the inherent characteristics of the FPGA architecture, it can perform massively parallel, real-time processing. Inline FPGA accelerators can be leveraged to handle streaming processing tasks such as early stage processing including initial filtering, transformations, data cleaning, and enrichments. The FPGA inline accelerators reside between the network interface card (NIC) and the CPU. These accelerators intercept the incoming packets going from the NIC to the CPU. With incoming streaming data into the FPGAs, the data typically gets split up to be processed in parallel. Depending on the application, the data can feed into different functions for pre-processing and/or reduction in real time. This may include transformations, pattern detectors, filters, encryption/decryption, and compression/decompression functions. These processing functions on the streaming data can provide real-time invaluable insights in data analytics, along with reducing the data volumes before passing through many layers of software stacks associated with big data frameworks. This can also significantly reduce the workload sent downstream, hence offloading the CPU.

The key advantage of the FPGA is the capability to directly ingest the data, while performing inline processing and/or data reduction. The FPGA can provide low-latency interface for data ingestion and inline processing capabilities before actual data movement to storage systems. This capability can significantly decrease processing latency that is a requirement for mission-critical applications.

In addition, the FPGA-based accelerators approach is deterministic regardless of data rate or data formats. This simplifies the system by eliminating the need for complex flow control and load balance management [19]. Figure 2 shows the inline accelerator data flow block diagram using FPGA accelerators.

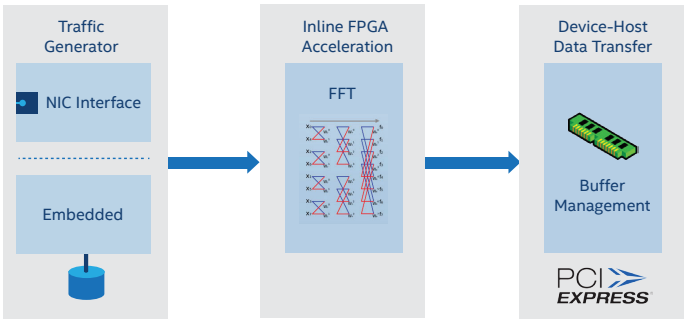

Figure 2. Inline Acceleration Data Flow Block Diagram

Hardware Implementation and Methodology

The design demonstrates low-latency ingestion of data using the FPGA, along with pre-processing operation. The data ingestion is done using 10GbE interface and UDP protocol. The Intel FPGA accelerator is based on Nallatech 385A card and includes an Intel Arria 10 FPGA. The card contains dual cage for enhanced small form factor pluggable (SFP+) interface, which is populated with 10G SFP+ SR optical module. The data arrives over optical link and is received by the FPGA. The FPGA accelerator is implemented using the Intel FPGA SDK for OpenCL framework, and the data is provided to the OpenCL kernels directly using pipes capability of OpenCL. The FPGA processing kernel system implements the fast Fourier transform (FFT) pre-processing and detection as an example of typical workload on input data. After inline processing is completed, the application transfers the results to the CPU host for further downstream processing.

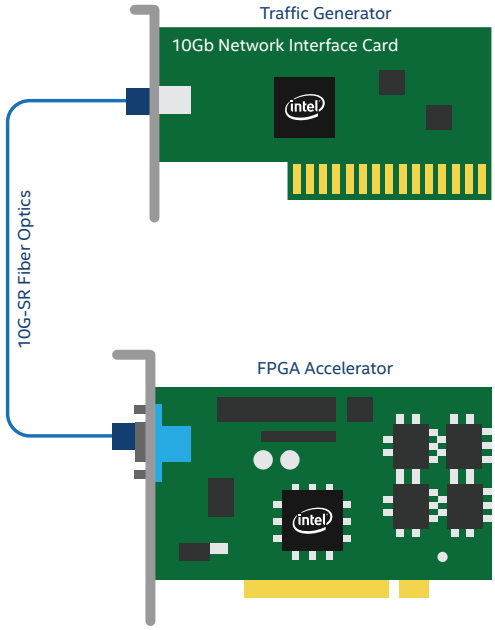

Figure 3. Hardware System Components

Traffic Generator

The traffic generator client code is built in C on a Linux* platform using the network programming model. The UDP sockets are not connection oriented. Hence, on the UDP client, it does not connect to a server. The program flow can be simplified as:

socket() -> sendto()

To improve the traffic data rate and maximize the throughput, the multithread programming model is used with POSIX thread (pthread) libraries. This allows spawning of multiple concurrent processes on the multiprocessor system, gaining speed through parallel processing. The UDP frames are transmitted using an NIC, Intel 82559 chipset, over 10GbE connection to the FPGA accelerator card.

FPGA Accelerator

The FPGA accelerator is implemented using the Nallatech 385A Intel Arria 10 FPGA card using Intel FPGA SDK for OpenCL for FPGA flow. This flow uses the board support package (BSP) with 10GbE interfaces and host pipes enabled. Figure 4 shows the block diagram of the FPGA accelerator implementation.

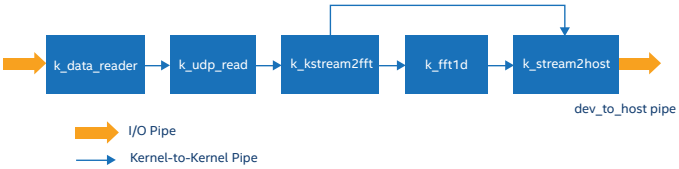

Figure 4. FPGA Accelerator Block Diagram Implemented Using OpenCL

The BSP provides the raw frame received over 10GbE interface to the OpenCL framework using I/O pipes. The first two kernels implement the network interface logic. The k_data_reader0 kernel extracts the UDP payload after pre-filtering the data based on the IP header information. The k_udp_read kernels reformats the data into single precision floating-point numbers before sending them through the kernel-to-kernel channels for downstream processing. The kernel-to-kernel channels are efficient in streaming data from one block to the next within the FPGA without involving the host or data transfer to/from external memory. The FFT accelerator consists of two OpenCL kernels. The k_stream2FFT kernel temporarily stores the streaming data and reorders the sequence for the actual FFT engine. The k_fft1d kernel implements the FFT engine using a parallel architecture. The final k_stream2host kernel sends the FFT results back to the host using the streaming host pipes. The streaming pipes are implemented similar to FIFOs or queues to allow streaming of results to the host from the FPGA kernel via PCIe, without involving data transfers using external memory.

Implementation Results

The performance of the system is measured using two separate metrics—latency and sustainable data rate. The benchmark compares the FPGA inline acceleration versus an all software implementation on the CPU, without a hardware accelerator. The FPGA setup uses the Intel Arria 10 FPGA. The CPU setup uses the Intel i7-7700K CPU at 4.2 GHz with eight logical cores. In both setup, the same traffic generator is used to generate the UDP packet data stream.

Latency

The latency measurement starts from the point when the UDP packets are sent out from the traffic generator and ends when the FFT results are received back on the host. Figure 5 shows the setup for latency measurements for the FPGA inline acceleration.

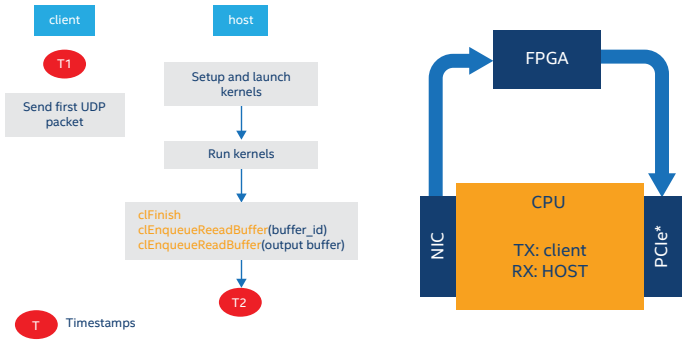

Figure 5. FPGA Inline Acceleration Setup for Latency Measurement

Figure 6 shows the setup for latency measurements for all the software implementation running on the CPU, without a hardware accelerator. UDP packets are transmitted via the NIC on a primary port and loops back to the secondary port on the same NIC via an optical link. The server uses the Data Plane Development Kit (DPDK) networking framework for fast packet processing that is optimized for Intel devices. The software implementation uses the FFT interfaces supported by the Intel Math Kernel Library (Intel MKL), which provides highly optimized and extensively threaded routines. The setup uses the Intel 64 architecture, 64 bit interface layer, and the OpenMP* threading layer.

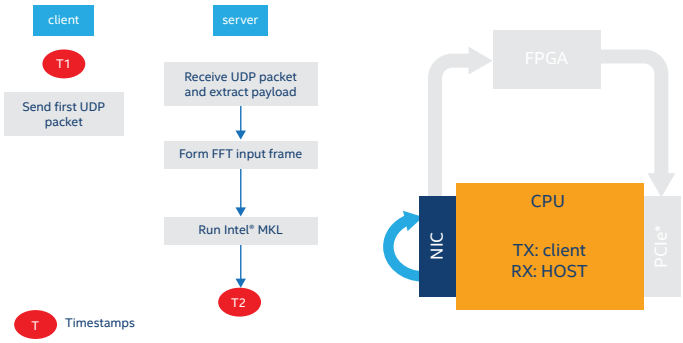

Figure 6. CPU Setup for Latency Measurement

The latency in traditional NIC->CPU->MKL processing pipeline is up to 22X longer compared to the data ingestion and inline processing in FPGA while the data is in motion.† Figure 7 shows the comparison of the latency for both setups running FFT inline acceleration.

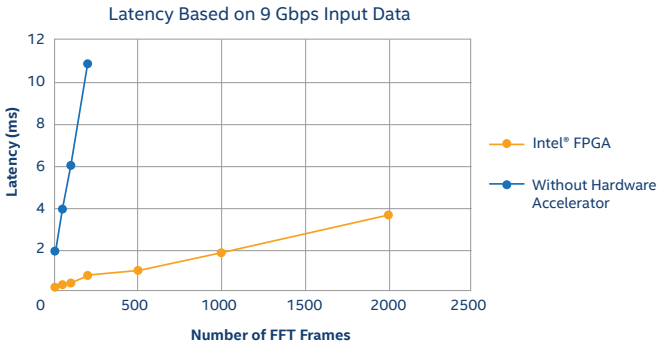

Figure 7. Comparison of Latency Measurement for FFT Inline Acceleration

Sustainable Data Rate

In the FPGA inline acceleration setup, the drop packet rate (DPR) is close to 0 based on the traffic generator data rate of 9 Gbps. The DPR is calculated based on the number of packets received where data ingestion is running in parallel with the inline processing of FFT. Without the FPGA hardware accelerator, the DPR reached >80% as the traffic generator data rate approaches 9 Gbps. This is a result of compute power starvation on a given CPU while executing traffic generation, networking stack management, and Intel MKL processing. By offloading the FFT transform inline processing to FPGA, the system is balanced and can keep up with incoming data and processing requirements. Figure 8 shows the DPR without the FPGA inline accelerator.

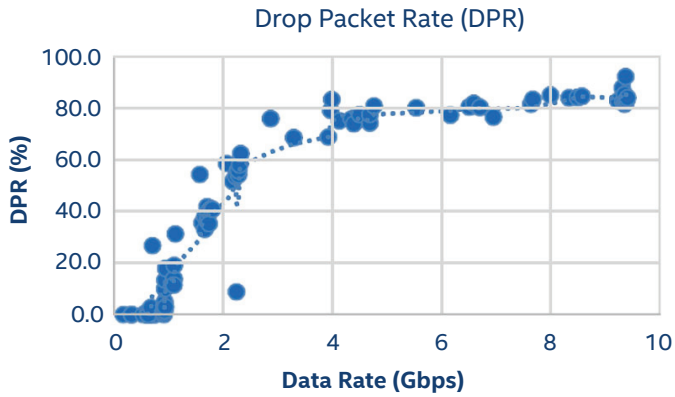

Figure 8. Drop Packet Rate Without Inline Acceleration

Summary

The FPGA inline accelerator provides a complementary approach to the existing streaming analytics ecosystems. FPGA can perform data ingestion in streaming analytics and inline processing in parallel to handle streaming processing tasks, such as filtering, transformation, data cleaning, and enrichment in real time. This can significantly reduce the workload and volume of data sent downstream, hence offloading the CPU. The FPGA inline accelerator offers multiple key advantages including low-latency interface. This is critical to data analysis applications that detect anomaly conditions and in return generate timely alerts. The benchmark results show that the FPGA inline accelerator has 22X lower end-to-end latency while sustaining a data rate of 9 Gbps with no drop packets compared to a system without a hardware accelerator.†

References

[1] "Why streaming analytics?", https://www.flexaware.net/why/streaming-analytics/

[2] "Real-Time Analytics: Streaming Big Data for Business Intelligence", https://dzone.com/articles/real-time-analytics- streaming-big-data-for-busines

[3] "Streaming Analytics 101: The What, Why, and How", http://www.dataversity.net/streaming-analytics-101/

[4] "WHITE PAPER CLOSING THE REAL-TIME INTELLIGENCE GAP", January 2015, https://marketing.napatech.com/acton/attachment/14951/f- 003f/1/-/-/-/-/Napatech_White_Paper_Closing_the_Real_Time_Intellige nce_Gap.pdf

[5] S. Chalamalasetti, M. Margala, W. Vanderbauwhede, M. Wright and P. Ranganathan, "Evaluating FPGA-acceleration for real-time unstructured search," 2012 IEEE International Symposium on Performance Analysis of Systems & Software, New Brunswick, NJ, 2012, pp. 200-209.

[6] Anshuman, Verma, Helal, Ahmed E., Krommydas, Konstantinos, and Feng, Wu-chun, "Accelerating workloads on FPGAs via OpenCL: A case study with OpenDwarfs," Computer Science Technical Reports, 2016.

[7] D. Chen and D. Singh, "Invited paper: Using OpenCL to evaluate the efficiency of CPUS, GPUS and FPGAS for information filtering," 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, 2012, pp. 5-12.

[8] Amir Momeni, Hamed Tabkhi, Yash Ukidave, Gunar Schirner, and David Kaeli. 2016. Exploring the Efficiency of the OpenCL Pipe Semantic on an FPGA. SIGARCH Comput. Archit. News 43, 4 (April 2016), 52-57.

[9] Nicholas Weaver, Vern Paxson, and Jose M. Gonzalez. 2007. The shunt: an FPGA-based accelerator for network intrusion prevention. In Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays (FPGA '07). ACM, New York, NY, USA, 199-206.

[10] D. Chen and D. Singh, "Fractal video compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs as acceleration platforms," 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC), Yokohama, 2013, pp. 297-304.

[11] S. Denholm, H. Inoue, T. Takenaka, T. Becker and W. Luk, "Low latency FPGA acceleration of market data feed arbitration," 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors, Zurich, 2014, pp. 36-40.

[12] T. Kenter, J. Förstner and C. Plessl, "Flexible FPGA design for FDTD using OpenCL," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, 2017, pp. 1-7.

[13] B. Sukhwani et al., "Database Analytics: A Reconfigurable-Computing Approach," in IEEE Micro, vol. 34, no. 1, pp. 19-29, Jan.-Feb. 2014.

[14] "Machine Learning on Intel FPGAs", https://software.intel.com/en-us/articles/machine- learning-on-intel-fpgas

[15] "Altera Implementing FPGA Design with the OpenCL Standard", https://www.altera.com/en_US/pdfs/literature/wp/wp- 01173-opencl.pdf

[16] Kang, K. and Peter Yiannacouras. "Host Pipes: Direct Streaming Interface Between OpenCL Host and Kernel." IWOCL (2017).

[17] Hamid Reza Zohouri, Naoya Maruyama, Aaron Smith, Motohiko Matsuda, and Satoshi Matsuoka. 2016. Evaluating and optimizing OpenCL kernels for high performance computing with FPGAs. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16). IEEE Press, Piscataway, NJ, USA, Article 35, 12 pages.

[18] "Intel® FPGA SDK for OpenCL™ Custom Platform Toolkit User Guide, Updated for Intel Quartus® Prime Design Suite: 17.1", https://www.altera.com/en_US/pdfs/literature/hb/ opencl-sdk/ug_aocl_custom_platform_toolkit.pdf

[19] "Accelerate KAFKA Producers with FPGAs", http://www.nallatech.com/rate-format-accelerating-kafka-producers-fpgas/

† Tests measure performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.