International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1S3, December 2019

Microblaze-Based Coprocessor for Data Stream Management System

Tareq S. Alqaisi, Linknath Surya Balasubramanian, Avinash Yadav, John J. Lee

Abstract: ***Data generation speed and volume have increased exponentially with the boom in Internet usage and with the advent of Internet of Things (IoT). Consequently, the need for processing these data faster and with higher efficiency has also grown in-over time. Many previous works tried to address this need and among them is Symbiote Coprocessor Unit (SCU), an accelerator capable of providing speedup of up to 150x compared with traditional data stream processors. The proposed architecture aims to reduce the complexity of SCU, making it flexible and still retaining its performance. The new design is more software driven and thus is very easy to be altered in the future if needed. We have also changed the older interface to industrial standard PCIe interface and AMBA AXI4 bus interconnect in order to make the design simple and open for future expansions.***

Keywords: Symbiote Coprocessor Unit, PCIe, AMBA AXI4.

I. INTRODUCTION

As there is an ever-increasing need for processing and storing data, many data stream processing systems have been developed to facilitate this process. Most of these works such as Aurora [1], Gigascope [2], and Nile [3] solve the problem through software-based approach using a host computer. However, SCU [4] tries to accelerate data stream processing with a hardware co-processor implemented in an FPGA. Heart of SCU [4] is the Single Instruction Multiple Data (SIMD) Very long Instruction Word (VLIW) execution engine that is coupled with a high bandwidth streaming memory system. The streaming memory system consisting of Streaming Register Files (SRF) can read or write multiple words per clock cycle. The SIMD-VLIW execution engine is monitored and controlled by a very complex state machine called as Stream Controller (SC). Although SCU shows a good improvement in performance of 12.3x to 150x compared with conventional DSMS processors, the hardware is very complex. Other limitation of SCU is its use of the old Hyper Transport (HT) interface. The above mentioned drawbacks of SCU are overcome in this study by (i) having a coprocessor like Microblaze [5] that substitutes the role of Stream Controller, (ii) replacing HT interface [6] with high speed PCIe [7] bus, and (iii) using AMBA AXI4 [8] bus interconnect instead of multiple proprietary bus interconnects. One of the many advantages of the new design is that since this Microblaze is coded in software, it is very easy to change the design for future and therefore more flexible.

Manuscript published on 30 December 2019.
* Correspondence Author (s)
Tareq S. Alqaisi, Department of Electrical and Electronics Engineering, Indiana University Purdue University Indianapolis, Indianapolis, USA. Email: tarek.alqaisi@gmail.com
Linknath Surya Balasubramanian, Department of Electrical and Electronics Engineering, Indiana University Purdue University Indianapolis, Indianapolis, USA. Email: linknath.ls@gmail.com
Avinash Yadav, Department of Electrical and Electronics Engineering, Indiana University Purdue University Indianapolis, Indianapolis, USA. Email: linknath.ls@gmail.com
John J. Lee, Department of Electrical and Electronics Engineering, Indiana University Purdue University Indianapolis, Indianapolis, USA.
Senior Member, ACM and IEEE. Email: johnlee@iupui.edu
URL: http://www.engr.iupui.edu/~ johnlee/

II. SYMBOTE COPROCESSOR UNIT (SCU)

This section briefly describes a previous work that we improved in this study. To use SCU, the queries are first written in query language called SymQL [4] and then given to a dedicated compiler which transforms these queries into executable kernel binaries. These binaries are then downloaded to Instruction Memory (IMEM). Next, the input streams are stored into the SRF. The SIMD-VLIW execution engine makes use of instruction and data level parallelisms to process these streaming input data with the direction from SC. Once the data has been processed, the results are then temporarily stored in SRF and given back to host processor. The overall architecture of SCU is shown in Fig. 1. The Hyper Transport Interface (HTI) at the left of the diagram connects SCU to host processor. HTI makes other input and output components such as Command/Response FIFOs and Read/Write stripes (WSP/RSP) visible to host processor. The host processor issues commands to SC through Command FIFO and sends data to SRF through WSP. As there are many clients waiting to read/write data from/into SRF, each one of them is given access to SRF through a round robin arbiter. Once the data is read by input FIFOs (IPF0 and 1) from SRF, the command for corresponding data is given by SC when this data comes into EU. The EU then computes the output and loads them into output FIFOs (OPF0-2) which are written to SRF. Once the whole query is processed, the SC generates an interrupt to host processor through Response FIFO, indicating the completion of processing, and then the results are sent to host processor through RSP when it acknowledges the interrupts and accepts the results.

A. Hyper Transport Interface (HTI)

The Hyper Transport Interconnect has four primary components namely HT Read Engine (HT-RDE), HT Write Engine (HT-WRE), HTI Read Interconnect (HTI-RDX), and HTI Write Interconnect (HTI-WRX). The read requests are queued in HT-RDE and are dispatched to the respective clients through HTI-RDX. The requests are also saved in an ordering queue to match the response from SCU.

Once the response is available, it is matched with its request ID and sent to host processor. The write requests from host are queued in HTI-WRE and dispatched to clients via HTI-WRX.

B. Stream Controller (SC)

Stream Controller (SC) acts as a proxy to the host

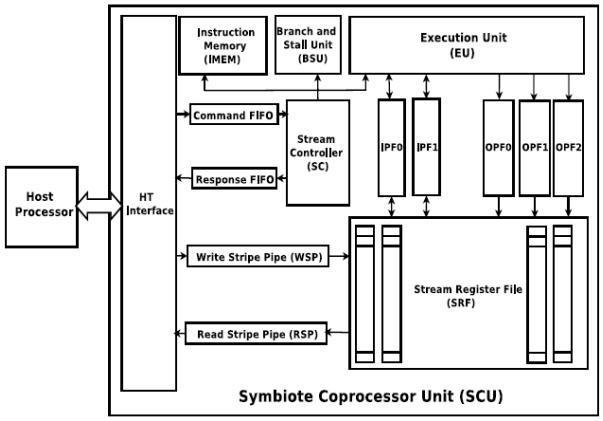

Fig. 1. Symbiote Coprocessor Unit architecture (Courtesy of Vaidya [4])

processor. The commands sent by host processor arrive to SC via HTI-WRE and are then decoded and executed by it. Three main components of SC are Stream Descriptor File (SDR), Kernel Descriptor File (KDR), and Offset Register File (ORF). SDR stores information such as start/end addresses of streams, tuple size, and tuple count of the streams in SRF so that the EU can utilize this information to fetch or store the correct and all required data from or into SRF. KDR has the information regarding kernel binaries in IMEM which have the commands to be executed on the data. ORF contains the information generated by SymQL compiler to optimize the execution of kernels.

C. Command/Response FIFOs (CRFIFO)

Requests and commands from host processor are sent to SC via command FIFO that obtains its input from HTI-WRE. Command FIFO has three signals named valid, command, and accept between itself and the SC. All responses of SCU to host processor are sent to HTI-RDE through response FIFO. It has write-enable, data, and accept signals between itself and SC.

D. Stream Register File (SRF)

SRF is the stream/tuple storage unit of SCU. It can serve multiple clients such as input FIFOs, output FIFOs, WSP, and RSP with a round robin arbiter. It has eight dual port memory banks for a total of 512KB. Each bank has two register files, namely even and odd banks. This architecture allows a double word to be written or read between host processor and SRF per clock cycle. The data are interleaved between eight banks such that EU can read or write eight words of data through input/output FIFOs in a single clock cycle.

III. PROPOSED CHANGE OF HARDWARE ARCHITECTURE

In general, the schema of tuples that are coming into the system does not change frequently but the data in each tuple change frequently. This property of stream gives way to a classification of the hardware that process tuples into two categories. One is the hardware unit that processes the data in every tuple. This unit is in a time critical path and is called as Stream Processing Unit (SPU) in the SCU architecture. The other is the hardware unit that configures the EU whenever the schema of input tuple changes. This unit is not required to be in the time critical path and is called as Stream Management Unit (SMU). Fig. 2 shows SPU, SMU, and the overall architecture of the new design. In redesigning of SCU, we are concerned that the performance remains approximately the same as before but now with a simpler design. To achieve this, we alter the SMU but do not touch SPU portion. The HT interface employed in SCU is obsolete at the time this study was being conducted, and no new FPGA was using HT interface; due to which we had to do one more alteration in the SCU, which is to change the HT interface to an industrial standard Gen3 PCIe interconnect, which increases the compatibility of our new design.

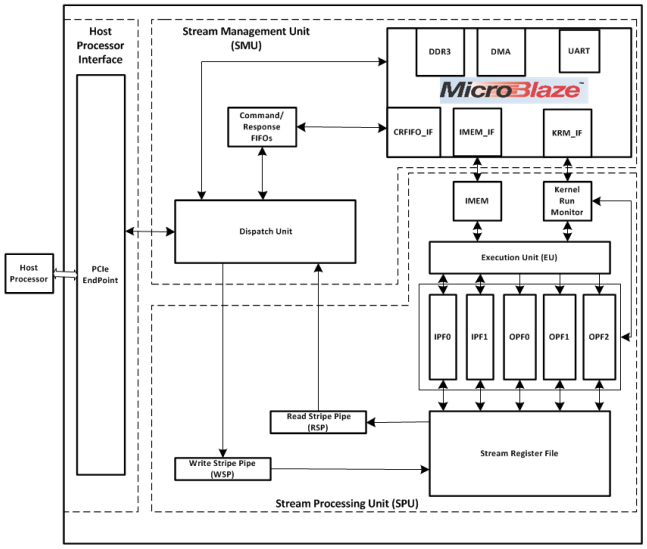

Fig. 2. MicroBlaze based coprocessing unit architecture

A. Peripheral Component Interconnect Express (PCIe)

PCIe is a standardized bidirectional computer expansion introduced by Intel in early 2000's. The protocol is based on point-to-point topology which connects two PCIe devices with links. Each link has one or more dedicated paths for sending and receiving data called as lanes. Each lane is capable of handling data up to 1GB/sec and there can be a maximum of 32 lanes per link in third generation (Gen3) PCIe. Each connected device exposes its internal memory to the host processor via a set of registers which represent the memory space of each of them. Such registers are called Base Address Registers (BAR). The host processor assigns a specific address in its own address space for each of these BARs and from then on, all transactions to that address are routed to its corresponding device.

B. Stream Management Unit (SMU)

The SMU consists of Microblaze [5] and its subsystem comprising Instruction Memory Interface (IMEM-IF), Command/Response FIFO Interface (CRFIFO-IF), Dispatch Unit (DU), Kernel Run Monitor Interface (KRM-IF), UART, and Direct Memory Access (DMA). Microblaze is a 32-bit RISC soft-core processor that runs at 125MHz with 32KB instruction and data caches along with 1MB of local memory and 4GB of DDR3 RAM. The Microblaze uses AXI4-lite to configure the peripherals and AXI4 for high throughput memory access. KRM-IF transfers kernel execution trigger signals from Microblaze to Kernel Run Monitor which in turn triggers the execution of operations in EU. KRM-IF has 4KB memory space to store KDR, SDR, and ORF. Whenever the Microblaze needs to know the status of a kernel or needs to start a new kernel, it sends request and command signals to KRM-IF through AXI4-lite. If Microblaze needs to access the internal memory of KRM-IF, it does through AXI4 interconnect. IMEM-IF exposes Instruction Memory to Microblaze, and it transforms the SCU specific Stream Controller Interconnect (SCTX) into AXI4-lite interconnect through a set of 32-bit memory-mapped registers. IMEM contains two internal buffers namely request and response buffers, and IMEM-IF exposes these buffers to Microblaze. Command/Response FIFO Interface (CRFIFO-IF) is responsible for transferring signals between CRFIFOs and Microblaze. It maps the commands sent by the host processor to two 32-bit read-only registers. CRFIFO-IF interrupts the Microblaze whenever there is a new command from host processor, and when the Microblaze is ready to accept the command, it notifies the CRFIFO through an accept signal, then the command is sent to Microblaze. DU exposes the address space of SPU to Microblaze and also provides a standard interconnect between SMU and PCIe interface.

IV. SOFTWARE ARCHITECTURE

Microblaze is a software-driven microcontroller that replaces the SC. This makes the new design easier to configure and control. The execution of kernels in our SCU is completed through four major steps: (i) configuration of coprocessor including downloading of kernels to execute, (ii) transferring input data from host to SCU, (iii) running the kernels, and (iv) reading back of results from SCU to host processor. The two software components that drive the hardware to perform above mentioned tasks are Stream Manager (SM) and peripheral drivers of CRFIF, DU, IMEM, and KRM.

A. Stream Manager (SM)

Stream Manager is the primary software component that has state machine to control the coprocessor. The five states of this state machine are idle state, command processing state, read kernel state, run kernel state, and configure kernel state. Initially, the machine is in idle state, and when a command arrives from host processor, it processes the command and goes to whichever state corresponding to the command. If the command is to configure the kernel, the KDR is set up, and then instructions and constants are sent to IMEM. If the system is in run kernel state, the Microblaze sends run kernel command to KRM via KRM-IF to trigger the execution of kernel and waits for its completion. If the system is in read kernel state, the corresponding KDRs are fetched and given to host.

B. Software Drivers

The KRM software driver facilitates the SM to communicate with KRM via KRM-IF. It has five functions such as enabling/disabling interrupts, reading kernel execution time, getting kernel status, and running kernel. Instruction Memory (IMEM) software interface driver makes the SM be able to read or write to IMEM via IMEM-IF. The interface first configures the IMEM with the number of instructions and constants to be stored in IMEM and then it transfers them to IMEM. The CRFIFO software driver allows SM to communicate with host processor through CRFIFO-IF. It interrupts the SM when a new command or data is received by CRFIFO, and once the SM is ready to accept it, they are transferred to SM.

V. EXPERIMENTATION AND RESULTS

To compare the performance of the new Microblaze-based coprocessor against the SCU, we executed three queries, namely map(n), filter(n), and aggregate(n). First, the host computer runs a program that configures the SMU. Then host computer stores the input tuples into SRF. Next, SPU executes the kernel and the results are read back by host processor from SRF. AXI4 hardware timer was added to Microblaze system to measure the execution time accurately. The timer starts to count when the KRM receives a run command and stops when KRM sends interrupt signal to Microblaze. The difference in execution times of both Microblaze-based coprocessor and SCU for running 4096 tuples and 16384 tuples, were recorded as 0.018ms for both. This delay is due to the handling of KRM interrupt by the Microblaze.

VI. SUMMARY

The proposed design made three changes in the SCU. First, a complex Stream Controller is replaced with a soft-core microprocessor named Microblaze. Second, an old HT interface is replaced with industry standard PCIe Gen3 interconnect. Finally, multiple proprietary interfaces used in SCU are replaced with AXI4 interconnection. The above-mentioned alterations to SCU make implementation, debugging, and testing easier for the new design. Other advantages of this new design is that it is very flexible, and thus, new algorithms and control logic can be more easily added to it in the future if required. Moreover, industrial standard AXI4 interconnect gives easier integration of new and existing AXI4 compatible peripherals, and the use of PCIe allows a wider selection of host processors.

REFERENCES

1. D. Abadi et al., “Aurora: A Data Stream Management System,” in *Proc. of ACM SIGMOD*, p.666, 2003.
2. C. Cranor, T. Johnson, O. Spatascheck, “Gigascope: A stream database for network applications,” *in proc. of ACM SIGMOD*, pp. 647-651, 2003.
3. M. A. Hammad et al., “Nile: a query processing engine for data streams,” in *proc. of ICDE*, p. 851, IEEE Computer Society, 2004.
4. P. S. Vaidya, *Hardware-software co-designed data stream management systems.* PhD thesis, Purdue University, West Lafayette, IN, USA, Dec. 2015.

5. H. Holden, J. Trodden, D. Anderson, “HyperTransport 3.1 Interconnect Technology.” http://www.mindshare.com/files/ebooks/HyperTransport%203.1%20Interconnect%20Technology.pdf, September 2008. Accessed: Dec. 2017.
6. Xilinx Inc, “MicroBlaze Processor Reference Guide.” http://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_1/ug984-vivado-microblaze-ref.pdf, April 2016. Accessed: Nov. 2017.
7. Arm Limited, “AMBA AXI and ACE Protocol Specification.” http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022d/index.html, October 2011. Accessed: Oct. 2017.
8. M. Jackson and R. Budruk, *PCI Express Technology*. MindShare, Inc., 2012.

AUTHORS PROFILE

Tareq S. Alqaisi, Embedded Systems Engineer, Master of Science, Department of Electrical and Computer Engineering, IUPUI, USA.

Linknath Surya Balasubramanian, Graduate Student, Department of Electrical and Computer Engineering, IUPUI, USA.

Avinash Yadav, Graduate Student, Department of Electrical and Computer Engineering, IUPUI, USA.

John J. Lee, Associate Professor, Department of ECE, IUPUI, USA. He received his PhD degree in Electrical and Computer Engineering from Georgia Institute of Technology. http://www.ece.iupui.edu/~johnlee/