A Power-efficient FPGA Accelerator: Systolic Array with Cache-coherent Interface for Pair-HMM Algorithm

Megumi Ito and Moriyoshi Ohara

IBM Research – Tokyo
19-21 Nihonbashi Hakozaki-cho, Chuo-ku, Tokyo 103-8510 Japan
E-mail: {megumii, ohara}@jp.ibm.com

Abstract: A systolic array is known as a parallel hardware architecture applicable to a wide range of applications. Naive implementations, however, can lead to inefficient resource usage and low power performance. In this paper, we discuss two techniques for improving the hardware resource usage: flexible multi-threading and dummy data padding. The design was implemented to accelerate a pair-HMM algorithm on an FPGA with the IBM POWER8 CAPI (Coherent Accelerator Processor Interface) feature. The CAPI feature simplifies the software design for driving the FPGA accelerator. Our experimental result indicates that the implemented FPGA accelerator executing the pair-HMM algorithm achieves 33x higher power performance than a POWER8 processor chip executing the same algorithm.

(Keywords: systolic array, accelerator, FPGA, and pair-HMM)

Introduction: As power efficiency is becoming a major concern in regard to computing, specialized accelerators are a promising approach to increase performance without increasing power consumption. To this end, FPGA is an attractive platform because it can be reprogramed for a specific workload dynamically when the demand rises and also because it is becoming to accommodate a large number of logic gates.

Taking full advantage of an FPGA for accelerating a workload is, however, not trivial. First, its clock frequency is typically an order of magnitude lower than that of current general-purpose processors. Accordingly, to accelerate its performance, an FPGA needs to exploit at least an order of magnitude more parallelism in a target workload than a CPU. Second, although an FPGA can potentially provide a significant performance advantage, it can take a significant effort to design an accelerator based on an FPGA with a large logic capacity. This implies that a modular design is necessary to manage the design complexity. Third, to access hardware devices based on FPGAs, a device driver with specialized API (Application Program Interface) is typically required, thereby imposing extra programming complexity. Since many software developers are already familiar with multi-thread programing, programmers should be allowed to use the same programming model for an FPGA accelerator without causing a significant performance overhead.

To overcome these challenges, in this paper, we examine how a systolic array architecture with CAPI can be applied to a real-world workload, namely a pair-HMM algorithm, which is used in genome-analysis applications for detecting variants in genome sequences. Because a single-threaded software implementation of a pair-HMM algorithm can take 51 hours to process a whole human genome [8], it is required to accelerate the algorithm so it can analyze a patient data set more quickly. A systolic array is a parallel architecture that can be applied to a wide range of problems in a modular manner to simplify the design complexity [1]. The CAPI technology [5] simplifies the software design by enabling an FPGA accelerator to access the main memory in a cache coherent way on IBM POWER8 systems [6]. Our evaluation has shown that our pair-HMM accelerator is 33x better than an 8-core POWER8 processor in terms of the power performance.

The contributions of this paper are: 1) a novel design of a systolic array for accelerating real-world applications, 2) an implementation of a pair-HMM accelerator based on that design, and 3) the improved power-performance of the proposed FPGA accelerator over a POWER8 processor chip.

Systolic Array Design: We designed a novel systolic array for computing a pair-HMM algorithm is proposed. It consists of a dynamic programming algorithm with a double-nested loop (Fig. 1) for calculating the matching probability of two genome sequences, namely, a read and a haplotype. The algorithm can be mapped to a one-dimensional systolic array as illustrated in Fig. 2, where each processing element (PE) performs a unit of computations (Fig. 3) in a pipelined fashion. This algorithm mapping is also applicable to the Smith-Waterman [3] and Viterbi algorithms [4]. When the problem size is larger than the systolic array size, it is divided into multiple sub-problems, which are then processed sequentially by using the output from a sub-problem as the input to the next sub-problem.

We introduced multi-threading to improve the throughput of the systolic array by extending the notion of

 978-1-5090-1386-9/16/$31.00 ©2016 IEEE

two-level pipelining [2]. As shown in Fig. 3, each PE has an internal data dependency, by which a previously computed Y_out is used to compute the next Y_out. The computation of the dependency path consists of one multiply (MUL) and one add (ADD), which takes 5 and 7 clock cycles respectively or 12 clock cycles in total. By taking advantage of fully pipelined MUL and ADD units on the FPGA, the proposed multi-threading architecture allows each PE to process 12 independent jobs in a pipelined fashion and generate an output at every clock cycle. Each thread represents a context of a single job. The threads are simply dispatched by round robin to avoid causing scheduling complexity and incurring a synchronization overhead between pipeline stages. Each thread can start and end processing a job independently from other threads. While a similar concept was introduced by Jacob et. al. [4], the proposed approach, namely, flexible multi-threading, reduces constraints among jobs executed in the same PE to further improve the utilization of pipeline stages.

When the last sub-problem size is smaller than the systolic array, we increase the problem size to the systolic array size by adding dummy input data, which is defined in such a way that they do not change the resulting probability value. This technique, dummy data padding, makes it possible to reduce the resource usage because a multiplexer for collecting the output from any stage of the systolic array can be omitted.

The overall structure of the pair-HMM accelerator is shown in Fig. 4. The Job Manager dispatches jobs to threads managed in the Thread Manager. Each Thread Manager controls the execution of a job and which data to be read from the Data Buffer and sent to the Systolic Array. The Data Buffer is used to store the input data from the main memory and the intermediate data generated by each sub-problem. For the last sub-problem, the Result block calculates the probability value, and the output is sent to the main memory.

Communication Design: Based on the IBM POWER8 CAPI [5], we designed a queuing mechanism for sending a job and its result between the CPU and the FPGA. Since the CAPI provides an FPGA accelerator with cache-coherent memory accesses, the accelerator can communicate and synchronize with a CPU thread in the exactly the same way as CPU threads do with each other. The proposed queuing mechanism uses a spin lock where the FPGA accelerator keeps checking the job queue until it finds a new job. Since the accelerator has a cache mechanism equivalent to that of the CPU, the memory bus traffic occurs only when the CPU writes a memory location to enqueue a new job. Thus, the cache-coherency support for the accelerator can significantly simplifies the software design for driving the accelerator.

Experimental results: The experimental environment is listed in Table 1. The pair-HMM accelerator on a Xilinx FPGA consumes 12.4 W according to a power consumption report by Xilinx Vivado SDK, while the 8-core POWER8 processor chip consumes 190 W at TDP [7]. The performance of the accelerator and a single-thread software implementation was measured in terms of throughput, namely, the number of millions cell updates per second (MCUP/s), as listed in Table 2. Because the throughput of the software implementation is expected to increase by 8x when all 8 cores are used, the throughput per processor chip is expected as listed in Table 2. Thus, the power-performance (MCUP per Joule) of the FPGA accelerator is 33x higher than that of the POWER8 processor.

Conclusion: We proposed a novel systolic array design to accelerate software applications with an FPGA. By applying the design to a pair-HMM algorithm, it was possible to increase power performance of an FPGA accelerator by 33x compared to that of a POWER8 processor.

References

[1] K. T. Kung, “Why Systolic Architectures?”, IEEE Computer, 15(1):37–46, January 1982.

[2] H. T. Kung, L. M. Ruane, and D. W. L. Yen, "A Two-Level Pipelined Systolic Array for Convolutions,", in VLSI Systems and Computations, Oct. 1981, pp. 255-264.

[3] K. Benkrid, Y. Liu, and A. Benkrid, “A Highly Parameterised and Efficient FPGA-Based Skeleton for Pairwise Biological Sequence Alignment”, IEEE Trans. on VLSI Systems, 17 (4), April 2009, pp. 561–570.

[4] A. C. Jacob, J. M. Lancaster, J.D.Buhler, and R.D. Chanberlain, “Preliminary results in accelerating profile HMM search on FPGAs”, IPDPS 2007.

[5] J. Stuecheli, B. Blaner, C. R. Johns, and M. S. Siegel, “CAPI: A Coherent Accelerator Processor Interface”, IBM Journal of Research and Development, vol. 59, no. 1, Paper 7, pp. 7:1–7:7, 2015.

[6] J. Stuecheli, "POWER8", Hot Chips, Aug. 2013.

[7] The Linley Group, “POWER8 Hits the Merchant Market”, 2014, accessed Feb. 2016 at http://www-03.ibm.com/systems//power/advantages/smartpaper/memorybandwidth.html

[8] Intel, “Replication Recipe for HaplotypeCaller Tool Runtimes”, 2015, accessed Feb. 2016 at http://www.intel.com/content/www/us/en/healthcare-it/solutions/documents/replication-recipe-haplotypecaller-tool-runtimes.html

 978-1-5090-1386-9/16/$31.00

```
for(i = 1; i < ReadLen; i++) {
  for(j = 1; j < HapLen; j++) {
    M[i][j] = (M[i-1][j-1] * Pmm[i] + X[i-1][j-1] * Pxm[i]
              + Y[i-1][j-1] * Pym[i]) * e[i-1][j-1]
    X[i][j] = M[i-1][j] * Pmx[i] + X[i-1][j] * Pxx[i]
    Y[i][j] = M[i][j-1] * Pmy[i] + Y[i][j-1] * Pyy[i]
}}
for(j = 0; j < HapLen; j++) {
  result += M[ReadLen-1][j] + X[ReadLen-1][j];
}
```

Fig. 1. Pair-HMM algorithm. *Pmm, Pxm, Pym, Pmx, Pxx, Pmy*, and *Pyy* are transition probabilities and *e* is emission probability. They are all constants in a given HMM model

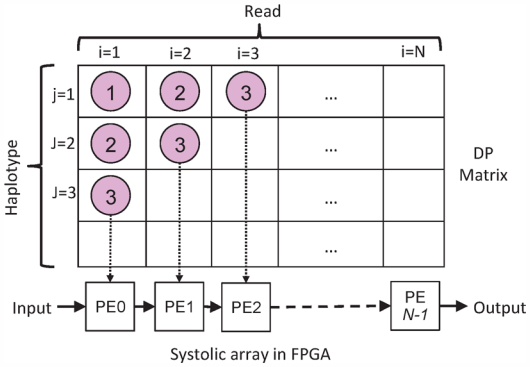

Fig. 2. Mapping a double-nested loop to a 1-D systolic array. Numbers in circles at (i, j) indicates the computation cycle at which PE(i-1) computes M, X, and Y at (i, j) of the nested loop in Fig 1. For example, at cycle 3, PE1 computes M[2][2], X[2][2], and Y[2][2].

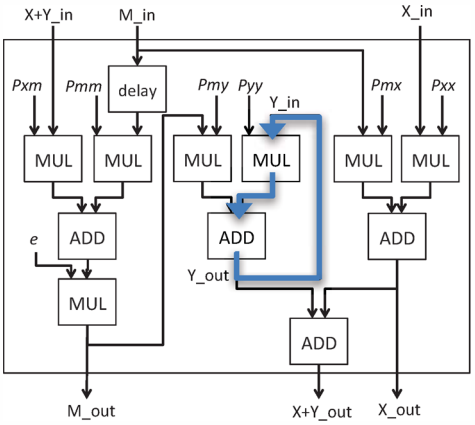

Fig. 3. Block diagram of a PE in the systolic array. X+Y is calculated before multiplying by *Pxm* because *Pxm* and *Pym* are the same in the implemented application. The loop marked with a thick line shows a data dependency within a PE consisting of a MUL and an ADD, which takes 12 clock cycles in total. The proposed PE impelments multi-threading, so it can process 12 independent jobs in a pipelined fashion and thus prevent the data dependency from reducing the resource utilization.

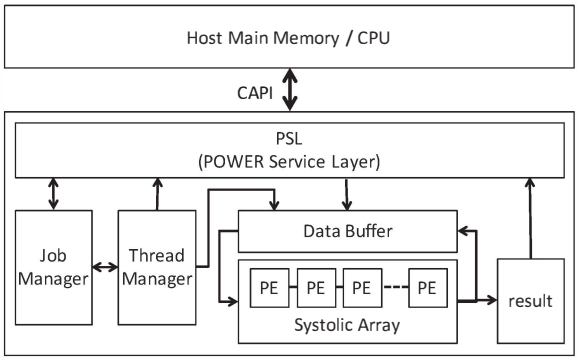

Fig. 4. Block diagram of the pair-HMM FPGA accelerator. The PSL enables the accelerator to access the main memory in a cache-coherent mode. The queuing mechanism between the accelerator and the CPU uses a spin lock for synchronization.

Table 1. Experimental environment.

CPU	IBM POWER8 8-core processor (8284-22A)	
FPGA	Xilinx Kintex UltraScale XCKU060-FFVA1156-2-e-es2	
	CLB LUTs	331,680
	Block RAM	38.0 Mb
	DSP Slices	2,760
Systolic Array	32PEs, 12 threads	
FPGA Resource Usage	CLB LUTs	46.5%
	Block RAM	42.6%
	DSP Slices	16.3%
Working Frequency	250MHz	
Input data	Chromosome 10 of the whole human genome dataset G15512.HCC1954.1.	

Table 2. Comparison of power performance of FPGA accelerator and POWER8 processor chip. Throughput is measured in Million Cell Updates per sec (MCUP/s)

	FPGA accelerator	IBM POWER8 processor
Power consumption	12.4W	190W
Throughput	1746 MCUP/s	809 MCUP/s (101 MCUP/s for a single-thread)
Power performance	140.95 MCUP per Joule	4.26 MCUP per Joule

POWER8 is a registered trademark of IBM Corporation. Other product and service names might be trademarks of IBM or other companies.

978-1-5090-1386-9/16/$31.00 ©2016 IEEE