3.3 Evaluation and Analysis

The integrated module configurations applying Algorithm 1 and Algorithm 2 are shown in Table 2 and Table 3, respectively.

Table 2: Integrated Module Configuration Applying Algorithm 1

Applications	Modules	Parameters	FPGA Configuration
K-means	Mapper	4 Dimension	Fully Configured
		6 Dimension	Fully Configured
		8 Dimension	Fully Configured
	Reducer	16 Centroid	Fully Configured
		32 Centroid	AES Decrypt 1/2
		48 Centroid	AES Decrypt 1/2
DNA sequencing	Mapper	16 Block, 6 String	Fully Configured
		16 Block, 8 String	Fully Configured
		16 Block, 10 String	Fully Configured
		32 Block, 6 String	AES Encrypt 1/2
		32 Block, 8 String	AES Encrypt 1/2
		32 Block, 10 String	AES Encrypt 1/2
		48 Block, 6 String	AES Encrypt 1/2
		48 Block, 8 String	AES Encrypt 1/2
		48 Block, 10 String	AES Encrypt 1/2
	Reducer	16 Block, 6 String	AES Decrypt 1/2
		16 Block, 8 String	Fully Configured
		16 Block, 10 String	Fully Configured
		32 Block, 6 String	AES Decrypt 1/3
		32 Block, 8 String	AES Decrypt 1/3
		32 Block, 10 String	AES Decrypt 1/3
		48 Block, 6 String	AES Decrypt 1/4
		48 Block, 8 String	AES Decrypt 1/5
		48 Block, 10 String	AES Decrypt 1/5

Table 3: Integrated Module Configuration Applying Algorithm 2

Applications	Parameters	FPGA Configuration
K-means	4 Dimension, 16 Centroid	AES Decrypt, Map 1/3
	4 Dimension, 32 Centroid	AES Decrypt, Map 1/5
	4 Dimension, 48 Centroid	AES Decrypt, Map 1/7
	6 Dimension, 16 Centroid	AES Decrypt, Map 1/3
	6 Dimension, 32 Centroid	AES Decrypt, Map 1/6
	6 Dimension, 48 Centroid	AES Decrypt, Map 1/9
	8 Dimension, 16 Centroid	AES Decrypt, Map 1/4
	8 Dimension, 32 Centroid	AES Decrypt, Map 1/8
	8 Dimension, 48 Centroid	AES Decrypt, Map 1/12
DNA sequencing	16 Block size, 6 Sub	Fully Configured
	16 Block size, 8 Sub	Fully Configured
	16 Block size, 10 Sub	Fully Configured
	32 Block size, 6 Sub	Fully Configured
	32 Block size, 8 Sub	Fully Configured
	32 Block size, 10 Sub	Fully Configured
	48 Block size, 6 Sub	Fully Configured
	48 Block size, 8 Sub	Fully Configured
	48 Block size, 10 Sub	Fully Configured

Fig .3 shows the hardware resource utilization of integrated mapper and reducer modules when applying Algorithm 1. In most cases, it is insufficient BRAM resources that cause a modification of the initial configuration. As the AES module occupies the majority of BRAM resources, it is split by Algorithm 1. The execution time of integrated mapper and reducer module for separate barrier scheme is shown in Fig .4 and it implies that *hmac* is the critical path of both target applications in all the cases of mapper. Because reducers do not contain *hmac* module, all the reducers are relatively faster than the corresponding mapper.

Fig .5 shows the hardware cost of integrated combined module with the configurations of Algorithm 2. Fig .6 shows the execution time of integrated combined module when applying combining scheme. Even not considering the network traffic between programmable SoC and Cortex-A9, the combining scheme shows quite remarkable speedup. Especially in DNA Sequencing, as assuming the number of utilizing boards is same, the highest speedup (293.08x) is achieved with the case of *16 Block size and 10 String size* in DNA Sequencing whereas the lowest speedup (2.20x) is achieved with the case of *8 dimension and 48 centroids* in K-means. The elimination of AES crypto module makes more MapReduce modules can be accommodated and the elimination of *hmac* module reduces the processing time of mappers dramatically.

Figure 3: Hardware resource utilization of integrated modules applying separate barrier

Figure 4: Execution time of integrated modules applying separate barrier

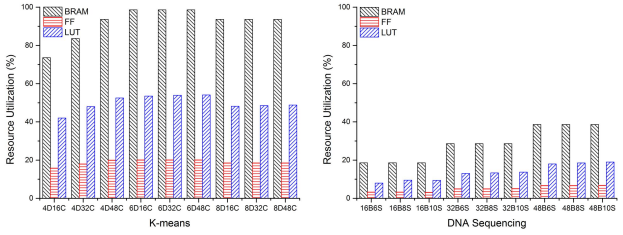

Figure 5: Hardware resource utilization of integrated modules applying combining scheme

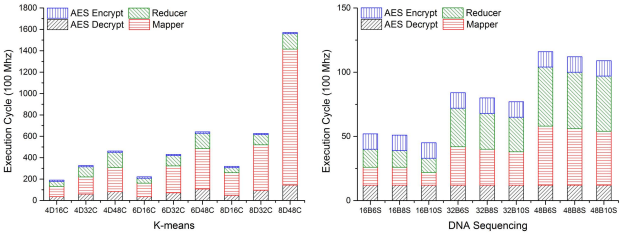

Figure 6: Execution time of integrated modules applying combining scheme

4 RELATED WORK

There have been efforts of accelerating MapReduce in two ways: the first takes advantage of off-the-shelf fixed parallel computing resources such as GPU, Xeon Phi, and many-core processors[6-10] and the second one utilizing reconfigurable hardware[11-13]. In the reconfigurable camp, Mershad et al[11] proposed a new service model where users can choose to pay a premium for faster data processing by exploiting FPGAs. Lin et al[12] proposed an eight Zynq-based Hadoop cluster, referred to as ZCluster. Shan et al[13] implemented a MapReduce framework on FPGA, referred to FPMR. In FPMR, the whole process of the mapper and reducer is conducted on the FPGA side, and the mapper and reducer are scheduled by hardware queue. There are a few studies on reducing the synchronization overhead of MapReduce by eliminating the global barrier[14, 15]. Elteir et al [14] proposed a hierarchical reduction, where the map and reduce processing is overlapped at the inter-task level and the reduce task gets started as soon as a certain number of map tasks complete. The partial outputs from reducers are aggregated following a tree hierarchy. Verma et al [15] classified reduce operations according to applications' characteristics, and eliminated the global barrier in Hadoop by using tree based scheme.

ACKNOWLEGEMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. 2017M3C4A7081956).

REFERENCES

[1] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the ACM. 2008;51(1):107-13.
[2] Crockett LH, Elliot RA, Enderwitz MA, Stewart RW. The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc: Strathclyde Academic Media; 2014.
[3] Feist T. Vivado design suite. White Paper. 2012;5.
[4] Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM.
[5] Krawczyk H, Canetti R, Bellare M. HMAC: Keyed-hashing for message authentication. 1997.
[6] Chen SY, Lai CF, Hwang RH, Chao HC, Huang YM, editors. A multimedia parallel processing approach on GPU MapReduce framework. Ubi-Media Computing and Workshops (UMEDIA), 2014 7th International Conference on; 2014: IEEE.
[7] Fang W, He B, Luo Q, Govindaraju NK. Mars: Accelerating mapreduce with graphics processors. IEEE Transactions on Parallel and Distributed Systems. 2011;22(4):608-20.
[8] Lu M, Zhang L, Huynh HP, Ong Z, Liang Y, He B, et al., editors. Optimizing the mapreduce framework on intel xeon phi coprocessor. Big Data, 2013 IEEE International Conference on; 2013: IEEE.
[9] Teodoro G, Kurc T, Kong J, Cooper L, Saltz J, editors. Comparative performance analysis of Intel (R) Xeon Phi (TM), GPU, and CPU: a case study from microscopy image analysis. Parallel and Distributed Processing Symposium, 2014 IEEE 28th International; 2014: IEEE.
[10] Honjo T, Oikawa K, editors. Hardware acceleration of hadoop mapreduce. Big Data, 2013 IEEE International Conference on; 2013: IEEE.
[11] Mershad K, Kaitoua AR, Artail H, Saghir MA, Hajj H, editors. A framework for multi-cloud cooperation with hardware reconfiguration support. Services (SERVICES), 203 IEEE Ninth World Congress on; 2013: IEEE.
[12] Lin Z, Chow P, editors. Zcluster: A zynq-based hadoop cluster. Field-Programmable Technology (FPT), 2013 International Conference on; 2013: IEEE.
[13] Shan Y, Wang B, Yan J, Wang Y, Xu N, Yang H, editors. FPMR: MapReduce framework on FPGA. Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays; 2010: ACM.
[14] Elteir M, Lin H, Feng W-c, editors. Enhancing mapreduce via asynchronous data processing. Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on; 2010: IEEE.
[15] Verma A, Cho B, Zea N, Gupta I, Campbell RH. Breaking the MapReduce stage barrier. Cluster computing. 2013;16(1):191-206.