Hardware Software Co-design of k-means Clustering Algorithm

Avadh Harkishanka
Electrical and Electronics Department
Birla Institute Of Technology and Science
Rajasthan, Pilani
avadh.h20180322@pilani.bits-pilani.ac.in

Amurt Prakash
Electrical and Electronics Department
Birla Institute Of Technology and Science
Rajasthan, Pilani
h20220254@pilani.bits-pilani.ac.in

Nikhil Gupta
Electrical and Electronics Department
Birla Institute Of Technology and Science
Rajasthan, Pilani
h20220253@pilani.bits-pilani.ac.in

Aatib Mohammad
Electrical and Electronics Department
Birla Institute Of Technology and Science
Rajasthan, Pilani
h20220239@pilani.bits-pilani.ac.in

Abhijit Asati
Electrical and Electronics Department
Birla Institute Of Technology and Science
Rajasthan, Pilani
abhijit_asati@ pilani.bits-pilani.ac.in

2023 9 I r d 979-8-3503-8320-1/23/$31.00 ©2023 IEEE I 10.1109/I 039 2023.10 1233

***Abstract*— The k-means clustering algorithm is a method that is frequently utilized for the purpose of grouping data points considering their similarity. Within the scope of this research, we investigate the viability of using a hardware-software co-design (HSC) strategy in order to speed up the k-means algorithm's execution. The studies are carried out using a Zedboard HSC platform based on Zynq 7000 architecture, which incorporates both processing system (PS) part implemented as set of instructions as software component and programmable logic (PL) part implemented on configurable FPGA fabric as hardware component using RTL code. In implementing k-means clustering algorithm, calculations of distance are carried out by PS and the results are communicated to PL, for performing the distance comparison & cluster reassignment. In order to reduce the resource utilization and the execution time, three different design configurations are being studied using HSC approach where the PL part follows different architectures. The results show comparison of execution speed, resource utilization and power when the different design architecture for the PL part are compared.**

Keywords—Zedboard, RTL, hardware-software co-design, PS/PL, BRAM, DDR GPIO, FSM, AMD Vitis, AMD Vivado, AVIRIS, ZYNQ 7000, ARM processor, NEON.

I. INTRODUCTION

Due to rising complexities in temporal and spatial computation, designers require to use mixed HSC style to improve the performance of the system. A pure hardware design is faster in operation but requires function specific silicon area due to custom build logical and memory structure in the system. Further, it is very complex to model and synthesize it since it requires to defines the behavioural and structural element of digital circuits. In contrast to pure hardware design, a pure software design requires pre-wired processor and memory to hold instruction which makes it programmable to perform any desired operation, thus offer flexibility in design and ease in development. The disadvantage is that software operations are much slower. Implementation of hardware is sometimes a brute-force approach to speeding up algorithm processing in many applications such as remote sensing [1], audio & video processing, image processing, and other high-speed signal processing such as in radar, artificial intelligence, machine learning, and many other applications specific integrated circuits (ASIC) and system on chip (SOC) solutions. Generally, the hardware implementation of a particular algorithm requires a lot more effort than its corresponding software implementation. Moreover, there are certain approximations and simplifications required to fit an algorithm into hardware [2]. The HSC provides a better alternative, to meet the requirements such as the design cycle time, cost reduction, testing time, power, performance, space, and timing complexities. In the HSC the software (PS part) is implemented as set of instruction to be executed on the processors (for the less time and power critical functions, which requires less design effort and hence offer flexibility) and hardware part implemented on PL using configurable FPGA fabric (for more time and power critical functions which requires more design effort and are less flexible). Hence the HSC optimizing area utilization, execution time, power and also offer flexibility. The integration of two different worlds (hardware and software) improves the functionality of the device. In the k-means clustering, a set of data points are separated into different clusters based on their similarities. It is a popular clustering algorithm because it is easy to implement and computationally efficient. Here, the software implementation is used for the calculations of distance and the distance comparison & cluster reassignment are performed in the hardware in order to speed up the calculations due to increased parallelism.

Since the calculated distance is taken in floating point format (IEEE 754 floating point representation) [3]. So, parallel

979-8-3503-8320-1/23/$31.00 ©2023 IEEE

comparators are designed in the hardware part to compare the distances given in the floating point numbers. There are different methods of distance calculation to cluster data set, like Euclidean and non-Euclidean metrics. These distance calculation techniques have different trade-off shown in [4] and their effects on total variance for some AVIRIS data [5]. Based on our data set, a simple Euclidean distance calculation method is used to measure the distance between data points and the centroid. This calculated distance is given to floating point comparators for the clustering process. Section II discusses the literature review, section III describes k-means clustering algorithm, next section IV discuss HSC approach, section V describes experimental setup, section VI described architecture partitioning in the experimental design, result, comparison and discussion is in section VII, finally section VIII concludes the paper.

II. Literature Review

Due to the recent interest in the k-means several applications, such as in image processing, the adaptive k-means algorithm is used for real-time video imaging [6]. Clustering algorithms like histogram-based algorithms [7], [8]; adaptive resonance networks-based clustering methods [8], [9]; fuzzy clustering algorithms [10], [11] proposed for video processing. These studies show how the pixel density and distance between pixels can be used to improve the image or video quality. Magnificent work is presented by Tarek S. Abdelrahman's [12] paper for the acceleration of the k-means algorithm, where the Intel Quick Assist Xeon+FPGA platform is used to implement the accelerated k-means algorithm on a tightly coupled CPU-FPGA System [12,13], which consists of "Arria 10 FPGA" and a multicore "Xeon processor". Both share access to memory using several physical channels which includes Intel's Quick Path Interconnect and two PCIe 3.0*8 links. In this research, we have used Zedboard HSC platform based on Zynq 7000 architecture, as explained later in section V.

III. K-means clustering algorithm

The k-means clustering algorithm is an unsupervised learning technique that is extensively used for the purpose of categorising data points into discrete groups based on their degree of resemblance to one another. The algorithm works in cycles as shown in Fig. 1, with the first step being a randomised selection of k-cluster centroids as the starting point. After that, a distance metric, such as the Euclidean distance, is utilised to determine which centroid each data point should be positioned within. The centroids are recalculated after the initial assignment is done by taking the mean of all of the data points that have been allocated to each cluster. *This process is repeated until convergence is achieved* (occurs when the centroids no longer move appreciably or when a maximum number of iterations is reached in the process). The clusters that were produced as a result indicate groups of data points that have similar patterns or attributes. k-means clustering has a wide range of applications in a variety of fields, such as customer segmentation, image compression, and anomaly detection.

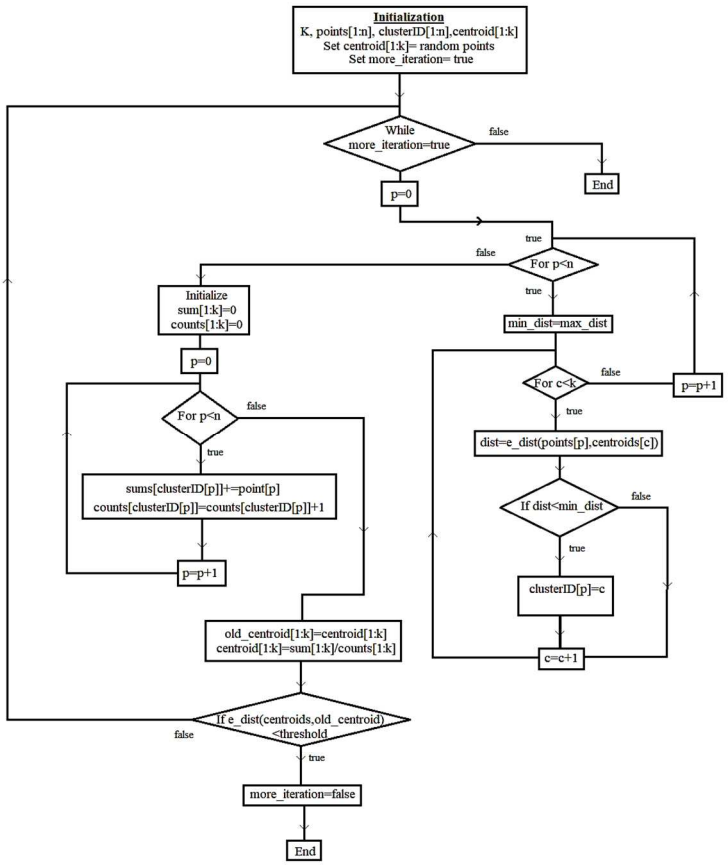

Fig. 1. Flowchart of k-means algorithm

IV. Hardware software co-design Approach

Hardware-software co-design is a collaborative approach that integrates both hardware and software components from the early stages of system development This methodology is particularly seen in the design of embedded systems, FPGAs, and system-on-chip (SoC) architectures [14]. It involves making strategic decisions about what tasks should be implemented in hardware and what should be handled in software, with the goal of finding the right balance to maximize system capabilities. This approach ensures that both the hardware and software components work seamlessly together, resulting in more efficient and cost-effective solutions for a wide range of applications, from IoT devices, machine leaning and high-performance computing systems. Hence, the advantages of HSC approach motivated us to implement K-means clustering algorithm and together with the prototyping flexibility of zynq-7000 architecture we tested three different designs each with their own end objective which is discussed further in the paper.

V. Experimental Setup

The experiments were carried out on the Zedboard HSC platform, which is based on the Zynq-7000 architecture, which combines PL and a PS. the PS part of Zedboard is a high-performance ARM processor that is integrated with PL hardware development. It includes a dual-core ARM Cortex-A9 MP Core processor with a NEON media processing engine, a DDR3 memory controller, an AMBA 4 AXI interconnect, and various other peripherals and interfaces. Xilinx Vivado design

suite was used for PL hardware development, while the Xilinx Vitis design tool suite (supports a range of operating systems, including Linux) was used for software implementation of the k-means algorithm. The software components were created using a high-level programming language and the APIs and libraries that were available for communication between the PS and PL. The PL part of the board provides flexibility and customizability for implementing hardware logic and interfacing with external devices. To facilitate data transfer between the PS and PL, we built communication channels such as GPIO ports or BRAM. We conducted comprehensive experiments with different datasets to evaluate the performance of our solutions, including key variables such as execution time, resource utilization, and clustering accuracy. We compared the results of each design architecture configuration to learn more about their performance features and trade-offs. The experimental arrangement ensured that the hardware and software components worked together seamlessly.

VI. Architecture Partitioning In Experimental Designs

An iterative strategy is used to conduct the k-means clustering algorithm in each design architecture configuration. The following steps were engaged in the iterative process as shown in Fig. 2:

(1) Initialization: The initial cluster centroids were calculated using a specified number of clusters and data points chosen at random.

(2) Distance Calculation: Using the supplied formula, the software portion (PS) calculated the Euclidean distances between each data point and the cluster centroids. These distances were saved in BRAM in order to communicate with the PL.

(3) Distance comparison: The distances were compared in the PL either in parallel blocks or by a controller finite state machine (FSM).

(4) Cluster Reassignment: Based on the comparison results, the PL then reorganized the data points into new clusters.

(5) Check for Convergence: With new cluster arrangement the software side (PS) then checked for convergence by looking at whether the centroid coordinates were constant or if the error function reached a minimum value. If convergence was not reached, the iterative procedure was restarted at step 2.

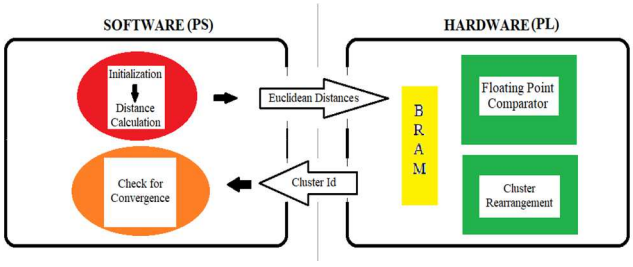

Fig. 2. Overview of the proposed framework for HSC

A. *Design architecture-1 (DA-1) flow PS and PL part and communication channels*

(i) PS Part: Firstly, based on the provided points, it creates the initial clusters (a, b) and (c, d). Then it determines the separations between every data point and the centroids. The distance values are then written to the BRAM.

(ii) PL Part: Firstly, it reads the BRAM's distance values. Here, four comparison IP (RTL) blocks are used in parallel to compare the distances between each data point. It produces the comparison data, which indicates which cluster each item belongs to. Finally, it rearranges the cluster and write back the new cluster arrangement corresponding to each point in the BRAM.

(iii) The PS component reads the comparison information from the BRAM: First it reads the new cluster arrangement from the BRAM storage device. Then it calculates the new centroid for each cluster. Further, begins the k-means algorithm's subsequent iteration if new centroid value calculated is not equal (or within some tolerance value) to previous centroid value.

Multiple distance comparisons can be processed simultaneously by employing parallel comparison RTL blocks in the PL portion, increasing the efficiency of the hardware design. This allows for faster computation and accelerates the clustering process. The block diagram of the DA-1 is shown in Fig. 3.

B. *Design architecture-2 (DA-2) flow PS and PL part and communication channels*

(i) PS Part: Firstly, utilizing the information provided, it creates the first two clusters, labeled (a, b), as well as (c, d). Then it determines the distances that separate each data point from the respective centroids. After that it saves current values of the distance to the BRAM.

(ii) PL Part: The controller FSM designed retrieves two distance values from the BRAM for each point simultaneously. When two distance values are fetched, they are sent for comparison meanwhile the FSM continues to read the other remaining distances, so overall all comparison can be done sequentially using one RTL comparison block. It produces the comparison data, which indicates which cluster each item belongs to. Then, Rearranges the cluster and write back the new cluster number corresponding to each point in the BRAM.

(iii) PS Part reads the updated cluster and iterates the algorithm: In this step, it reads the new cluster arrangement from the BRAM storage device. Then, calculates the new centroid for each cluster. Lastly, begins the k-means algorithm's subsequent iteration if new centroid value calculated is not equal (or within some tolerance value) to previous centroid value.

Here, we reduced the number of parallel comparison blocks that are needed by utilizing a controller FSM in the PL section which will properly control the fetching and writing of data to and from the BRAM also it will be responsible for feeding the data to the floating-point comparison RTL block sequentially for each pair it fetches. This will result in cost reductions associated with the utilization of hardware. However, it is essential to make certain that the controller FSM has been suitably constructed so that it can deal with the process of retrieving data, making comparisons, and writing data back to the BRAM, without any data conflicts and synchronization issue. The block diagram of the DA-2 is shown in Fig. 4.

C. *Design architecture-3 (DA-3) flow PS and PL part and communication channels*

The following is a rundown of the processes that went into our design architecture-3:

(i) PS Part: Firstly, based on the points that are provided, forms the initial clusters (a, b) and (c, d) respectively. Then it determines the distances that separate each data point from the respective centroids. Lastly, communicates the data regarding the distance directly to the PL side using the GPIO ports.

(ii) PL Part: Here, through the GPIO ports, it receives the distance data sent from the PS side. Then it performs a parallel comparison of all the distance data. Also produces the comparison data, which indicates which cluster each item belongs to. Finally, rearranges the cluster and write back the new cluster number corresponding to each point in the BRAM.

(iii) PS part reads the updated cluster and iterates the algorithm: It reads the new cluster arrangement via the GPIO port from PL side. Then it calculates the new centroid for each cluster. Also begins the k-means algorithm's subsequent iteration if new centroid value calculated is not equal (or within some tolerance value) to previous centroid value.

By bypassing the need for read/write operations in the BRAM and instead sending the distance data from the PS to the PL instantly, latency can be kept to a minimum. This could potentially increase the speed at which data is sent, which could boost your design's overall performance. However, data synchronization, signal integrity, and potential bottlenecks are all crucial to think about when communicating over GPIO ports due to the limited bandwidth and number of GPIO interfaces. This is because there is a certain amount of data transfer capacity for GPIO ports. Additionally, you must guarantee that the parallel comparison was properly designed on the PL side. The block diagram of the DA-3 is shown in Fig. 5.

E. *Floating Point Comparator*

The distance calculated is in floating-point format. So, to compare the distances between each data point and each centroid, we need a floating-point comparator that compares the first 8 bits of exponent (bit-30 to bit-23) and then compares the next 23 bits of mantissa (bit-22 to bit-0). Since the distance is always positive so sign bit is always “0”. The comparison block outputs the index of the nearest centroid, which is then written to BRAM.

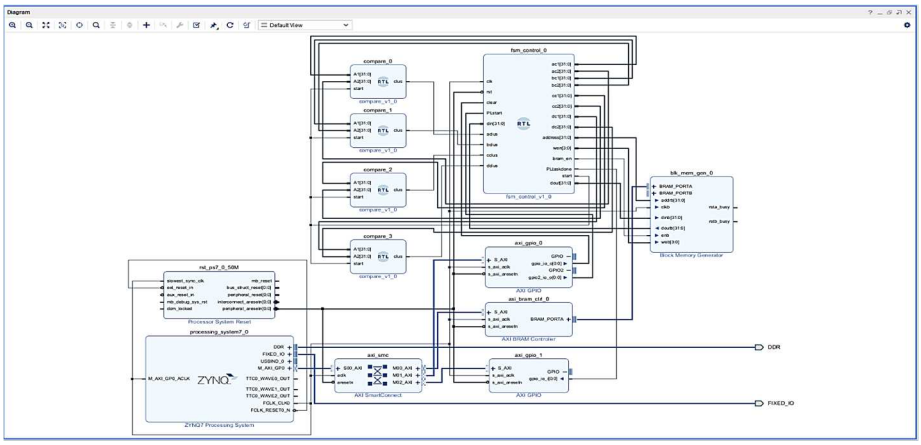

Fig. 3. DA-1 block diagram

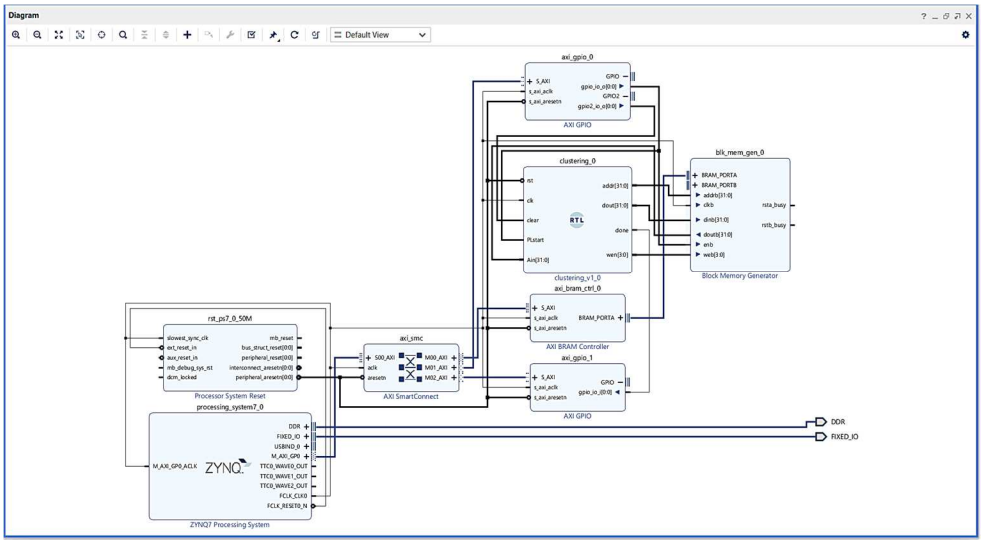

Fig. 4. DA-2 block diagram

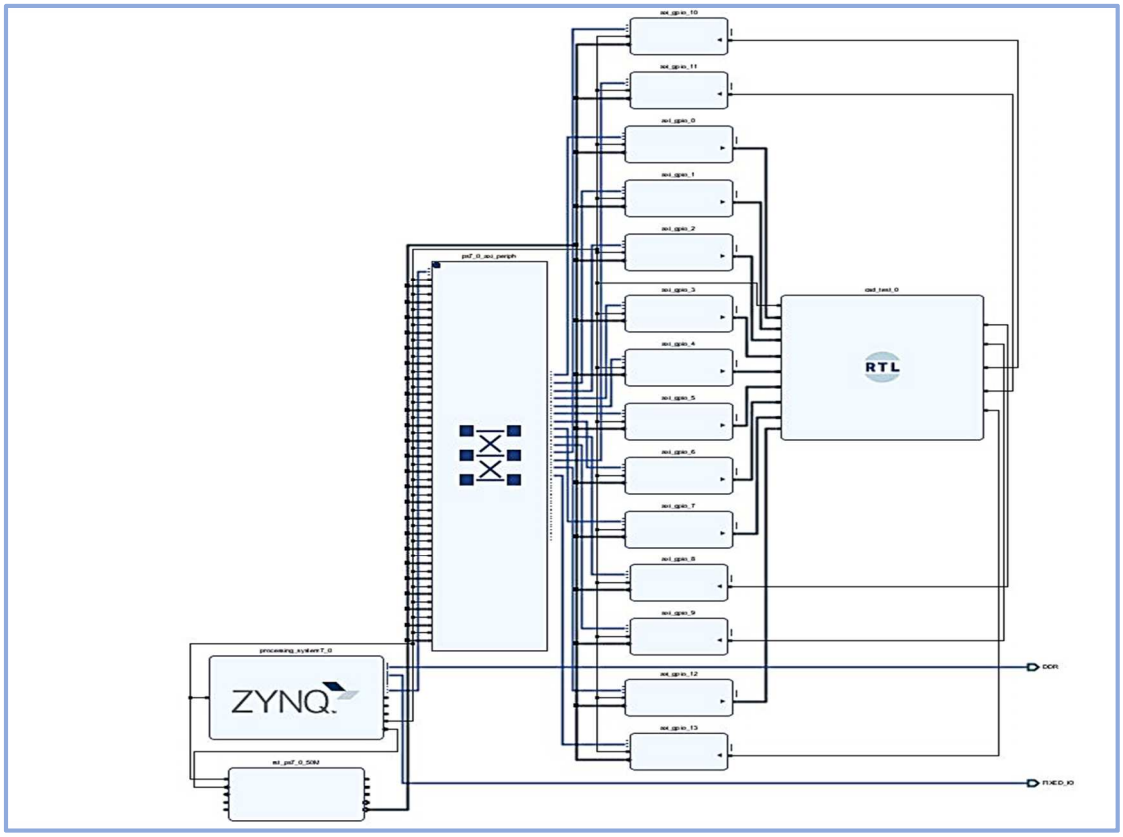

Fig. 5. DA-3 block diagram.

VII. Results , comparison and Discussion

We compared the three design iterations based on numerous metrics, such as runtime, memory usage, and clustering precision.

A. Design architecture-1 configuration: In the design architecture-1 configuration, four parallel comparison blocks were used in the PL to process the distance data concurrently. This method improved performance by making the most efficient use of available hardware to do comparisons of distances. However, this led to greater utilization of available resources. The utilization report obtained through synthesis is shown in the Table 1.

B. Design architecture-2 configuration: In design architecture-2, to evaluate two distance values concurrently, we implemented a controller FSM in the PL. In this method the systems resources are reduced by using fewer parallel comparison blocks. Overall processing time was comparable to design architecture-1, but scalability was improved. The utilization report obtained through synthesis is shown in Table 1.

C. Design architecture-3 configuration: In the design architecture-3 configuration, the distance data was transferred from the software (PS) to the hardware (PL) without using the BRAM. This architecture outperformed the other two alternatives in terms of execution speed by minimizing the impact of BRAM read/write latency. For Zedboard BRAM read latency saved is 15 clock cycles (15×20 ns = 300 ns) for each eight-distance data read. However, precise synchronization and awareness of any signal integrity problems were prerequisites. The utilization report obtained through synthesis is shown in Table 1. The accuracy of clustering i.e. results came out the same for each design architecture. Table 1 shows the timing results for each of design architecture-1, design architecture-2, and design architecture-3.

Table 1. Resources and Power Results.

Resource	Utilization			Availab-le	Utilization %		
	DA-1	DA-2	DA-3		DA-1	DA-2	DA-3
LUT	5060	5000	2216	53200	9.51	9.40	4.17
LUTRAM	732	732	83	17400	4.21	4.21	0.48
FF	5272	5087	3051	106400	4.95	4.78	2.87
BRAM	2	2	-	140	1.43	1.43	-
BUFG	1	1	1	32	3.13	3.13	3.13

As observed from Table 2, shows design architecture type (DA) worst negative slack for the setup violation (t_{setup}), worst hold slack (t_{hold}), worst propagation delay (t_p), number of clock cycles per iteration (NC), hardware execution time per iteration (T_{exe}), number of iteration (Ni), total execution time (T_{total}) ns.

Table 2. Timing Resulrts (Clock time period=20ns(50 MHz)

DA	t_{setup} (ns)	t_{hold} (ns)	t_p (ns)	N_C (per iteration)	T_{exe} (ns)	N_i	T_{total} (ns)	P_{avg} (W)
1	11.3	0.015	8.75	22	440	2	880	1.734
2	10.6	0.016	8.75	25	500	2	1000	1.734
3	11.9	0.022	9.02	3	60	2	120	1.713

Each design has no timing violation. Hardware execution time for design architecture-2 is more than design architecture -1 because as 4 comparators are replaced by single comparator, so sequential control through FSM takes slightly longer time. Hardware execution time for design architecture-3 is the least because as BRAM is removed so no more read/write latency and the FSM controller are also removed. When employing GPIO ports in design architecture-3 higher slack shows that the design has more timing margin, which means that it satisfies or surpasses the required timing limits. The execution time of design architecture-3 is much less as it doesn't read and write data from BRAM and directly accesses the data from GPIO ports. Calculations of distance are carried out by PS i.e. the software component, and the results are communicated to the hardware component in PL, for the distance comparison and cluster reassignment. *In the HSC implementation discussed in paper, the PS processor and memory resources are equal in all three architectures* further the PS part of each algorithm occupies nearly same instructions, while the PL part resources, speed and power results are architecture dependent and are compared in paper. The Table 2 describes the resources and power results of different design architectures for PL part, from the report generated by synthesis performed on Vivado hardware design tool. Each design consumes nearly the same average power. Design architecture-2 has fewer LUTS and FF counts as compared to design architecture-1, since architecture-2 uses a single floating-point comparator as opposed to four floating-point comparators used in architecture-1. The design architecture-3 consumes least hardware resources due to absence of time-consuming BRAM and also its corresponding FSM controller (required to synchronously fetch or write data into BRAM). Further the design architecture-3 also consumes least power as shown in Table 2, and the corresponding power-delay product is shown in the Fig. 6.

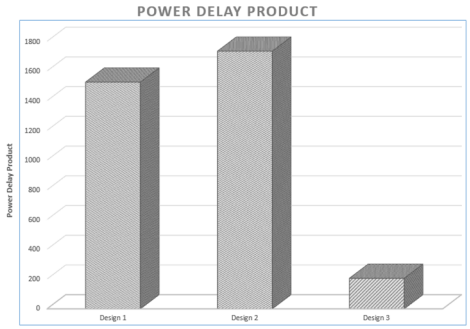

Fig. 6. Comparison of Power-Delay product

VIII. Conclusion

This research adds to the existing literature on k-means clustering algorithm implementations for three architecture alternatives, each with its own set of benefits and constraints using HSC approach. The design architectures using BRAM are slower and increases the operation times drastically. The research will help researchers and practitioners to select the best possible design strategy according to their unique needs and constraints, such as operation time, hardware resources, and power consumption. The investigation of the influence of these designs on large-scale datasets and the evaluation of their performance in comparison to the performance of other clustering methods would provide useful insights for real-world applications and can be the future scope of our work.

References

[1] Theiler, J. P., Frigo, J. R., Gokhale, M., and Szymanski, J. J., "Co-design of software and hardware to implement remote sensing algorithms", in Imaging Spectrometry VII, 2002, vol. 4480, pp. 86–99. doi:10.1117/12.453328.P. Huang, C. Wang, R. Ma, W. Liu, and F. Lombardi, "A Hardware/Software Co-design Method for Approximate Semi-Supervised

[2] P. Huang, C. Wang, R. Ma, W. Liu, and F. Lombardi, "A Hardware/Software Co-design Method for Approximate Semi-Supervised K-Means Clustering," 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, China, 2018, pp. 575-580, doi: 10.1109/ISVLSI.2018.00110."IEEE Standard for Floating-Point Arithmetic," in *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, vol., no., pp.1-84, 22 July 2019, doi: 10.1109/IEEESTD.2019.8766229.

[3] "IEEE Standard for Floating-Point Arithmetic," in IEEE Std 754-2019 (Revision of IEEE 754-2008), vol., no., pp.1-84, 22 July 2019, doi 10.1109/IEEESTD.2019.8766229.

[4] M. Leeser, J. Theiler, M. Estlick and J. J. Szymanski, "Design tradeoffs in a hardware implementation of the k-means clustering algorithm," Proceedings of the 2000 IEEE Sensor Array and Multichannel Signal Processing Workshop. SAM 2000 (Cat. No.00EX410), Cambridge, MA, USA, 2000, pp. 520-524, doi: 10.1109/SAM.2000.878063.

[5] Airborne Visible Infrared Imaging Spectrometer, 1999. http://makalu.jpl.nasa.gov/avi.ris.html

[6] B. Maliatski and O. Yadid-Pecht, "Hardware-driven adaptive k-means clustering for real-time video imaging," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 1, pp. 164-166, Jan. 2005, doi: 10.1109/TCSVT.2004.839977.

[7] M. J. Swain, "Interactive indexing into image databases," in *Proc. SPIE* Conf. Storage and Retrieval in Image and Video Databases, 1993, pp. 173–187.

[8] *G. Carpenter and S. Grossberg, "Invariant pattern recognition and recall by an attentive self-organizing art architxecture in a non-stationary world," in Proc. IEEE 1st Int. Conf. Neural Networks, vol. II, 1987, pp. 737–745.*

[9] T. Serrano-Gotarredona and B. Linares-Barranco, "A real-time clustering microchip neural engine," *IEEE Trans. VLSI Syst.*, vol. 4, no. 2, pp. 195–209, Jun. 1996.

[10] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," IEEE Trans. Pattern Anal. Mach. Intell., vol. 11, no. 10, pp. 773–781, Oct. 1989.

[11] E. L.Walker, "Combining geometric invariants with fuzzy clustering for object recognition," in Proc. Annu. Conf. North American Fuzzy Information Processing Society, New York, Jun. 1999, pp. 571–574.

[12] Abdelrahman, Tarek. (2020). Cooperative Software-hardware Acceleration of K-means on a Tightly Coupled CPU-FPGA System. ACM Transactions on Architecture and Code Optimization. 17. 1-24. 10.1145/3406114.

[13] Zubair, M., Iqbal, M.A., Shil, A. et al. An Improved K-means Clustering Algorithm Towards an Efficient Data-Driven Modeling. Ann. Data. Sci. (2022). https://doi.org/10.1007/s40745-022-00428-2.

[14] Rojas-Muñoz, L.F.; Rostro-González, H.; García-Capulín, C.H.; Sánchez-Solano, S. Hardware/Software Co-Design of a Circle Detection System Based on Evolutionary Computing. Electronics 2022, 11, 2686. https://doi.org/10.3390/electronics11172686.