Scalable and Parameterizable Processor Array Architecture for Similarity Distance Computation

Awos Kanan
Dept. of Computer Engineering
Princess Sumaya University for Technology
Amman, Jordan
a.kanan@psut.edu.jo

Fayez Gebali
Dept. of Electrical & Computer Engineering
University of Victoria
Victoria BC, Canada
fayez@uvic.ca

Atef Ibrahim
Dept. of Computer Engineering
Prince Sattam Bin AbdulAziz University
Al-Kharj, Saudi Arabia
aa.mohamed@psau.edu.sa

Kin Fun Li
Dept. of Electrical & Computer Engineering
University of Victoria
Victoria BC, Canada
kinli@uvic.ca

***Abstract*—Processor array architecture is a popular approach to improve computation of similarity distance matrices; however, most of the proposed architectures are designed in an ad hoc manner, some have not even considered dimensionality and size of the datasets. We believe a systematic approach is necessary to explore the design space. In this work, we present a technique for designing scalable processor array architecture for the similarity distance matrix computation. Implementation results of the proposed architecture show improved compromise between area and speed. Moreover, it scales better for large and high-dimensional datasets since the architecture is fully parameterized and only has to deal with one data dimension in each time step.**

***Index Terms*—Processor arrays, scalable hardware, similarity measures, parallel computing.**

I. Introduction

The computational complexity of machine learning and data mining algorithms, that are frequently used in today's applications such as embedded systems, makes the design of efficient hardware architectures for these algorithms a challenging issue. The computation of similarity distance matrices is one of the computation kernels that are generally required by several machine learning and data mining algorithms to measure the degree of similarity between data samples [1]. For several algorithms such as K-Means [2], SVM [3], and K-NN [4], distance calculation is considered a computationally intensive task that accounts for a significant portion of the processing time, especially when dealing with large and high-dimensional datasets [5].

Given the complexity of today's data, machine learning and data mining algorithms are expected to handle big and high-dimensional data. Linear [6] and 2-D [7] [8] processor array architectures have been proposed to accelerate the computation of similarity distance matrices. 2-D processor arrays are generally faster than linear arrays as more processing elements (PEs) are used to perform the computation in parallel. On the other hand, linear arrays are more suitable for area, power, and bandwidth-constrained applications. To the best of our knowledge, data dimensionality and size have not been considered in previous 2-D processor arrays. For high-dimensional data, feeding all the features of a single data element simultaneously is not practical due to I/O bandwidth constraints. Linear processor arrays, on the other hand, are generally more area-efficient and meets I/O bandwidth constraints at the cost of much slower execution time. In this paper, we present a systematic approach for developing scalable processor array architecture for similarity distance computation based on our recent work in [9] with more control on area and I/O requirements and better compromise between area and speed.

The rest of this paper is organized as follows: Distance computation problem is formulated in Section II. In Section III, the systematic technique that is employed to develop the processor array architecture is introduced. The proposed architecture is presented in Section IV. Design comparison and implementation results are presented in Section V and Section VI, respectively. Finally, Section VII concludes the paper.

II. Similarity Distance Computation

Given dataset $\mathbf{X}$ of N samples and dataset $\mathbf{Y}$ of K samples, where each sample in the two datasets has M features. A similarity measure such as Manhattan, Euclidean, or Cosine distance [1] [10] can be used to generate a distance matrix $\mathbf{D}$ of $K \times N$ elements. The distance between the n^{th} sample of dataset $\mathbf{X}$ and the k^{th} sample of dataset $\mathbf{Y}$ is represented by the value of element $D(k,n)$ of matrix $\mathbf{D}$. In this paper, Manhattan distance is used to illustrate the calculation of similarity distance matrix between data samples of the two datasets $\mathbf{X}$ and $\mathbf{Y}$. Manhattan distance can be expressed as:

$$D(k,n) = \sum_{m=0}^{M-1} |X(m,n) - Y(k,m)| \qquad (1)$$
$$0 \le k < K, \quad 0 \le n < N,$$

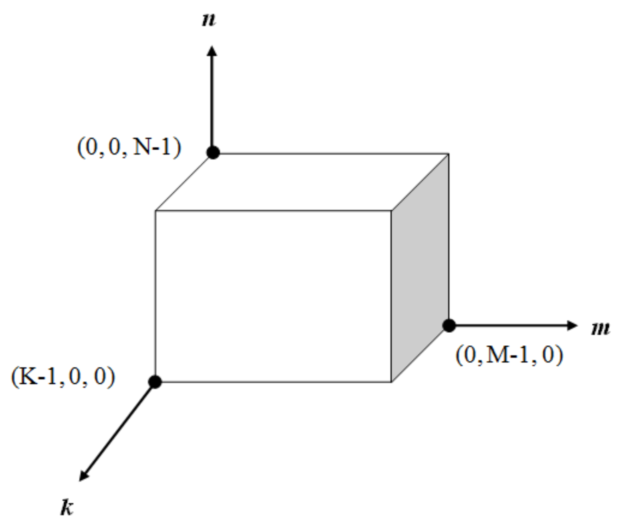

Fig. 1: Computation domain.

where N and K are the number of samples of datasets $\mathbf{X}$ and $\mathbf{Y}$, respectively, and M is the dimensionality (number of features) of the two datasets. The methodology employed to perform design space exploration in our work [9] relies on analyzing data dependencies using dependence matrices that show how input and output variables depends on indices k, m, and n [11]. The emphasis of this paper is on the parallelization of similarity distance computation regardless of the similarity measure used. Hence, the work presented in this paper can be generalized to other similarity measures.

Similarity distance computation in the K-Means clustering algorithm [2], for instance, is performed in the same way described in this section. Distances between N samples of dataset $\mathbf{X}$ and the set of centroids of K clusters $\mathbf{Y}$ are calculated to find the closest cluster for each data sample.

III. Methodology of processor array design

In our recent work [9], we have systematically explored the design space of 2-D processor array architectures for similarity distance computation using the methodology proposed by the second author in [12]. Six processor array architectures were obtained using linear scheduling and projection operations. In this work, nonlinear scheduling and projection are employed to develop a scalable and parameterized processor array architecture starting from one of the six obtained architectures that is chosen based on criteria discussed later in this section.

A. Computation Domain

As shown in Fig. 1, the computation domain $\mathcal{D}$ is defined by the three indices of the algorithm [11]. Every point in the computation domain has three coordinates that are represented as:

$$\mathbf{p} = \begin{bmatrix} k & m & n \end{bmatrix}^t \tag{2}$$

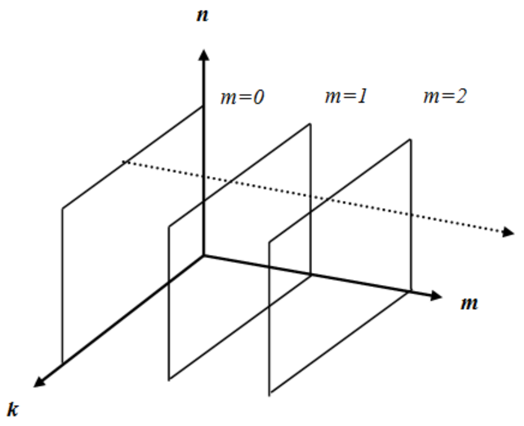

Fig. 2: Equitemporal zones for a linear scheduling function obtained in [9].

B. Data Scheduling

A scheduling function assigns each point in the computation domain a time value. Based on the choice of broadcasting or pipelining algorithm variables, we were able to obtain four linear scheduling functions in [9]. Linear scheduling can be used to perform design space exploration for the problem in hand. Depending on the values of K, M, and N, in order to obtain scalable architectures that are amenable for hardware implementation, more control on the computational load at each time step is required. Fig. 2 shows one of the obtained timing options for 2-D processor arrays. In this timing option, all points on any given plane with the same value of coordinate m are assigned the same time value and said to belong to a single equitemporal zone [11]. For large-scale data, having a large number of points executing at the same time results in an impractical hardware architecture that requires a huge number of PEs and the feeding of a large number of data inputs simultaneously. Nonlinear scheduling can be used for more control on the computational load to be performed at each time step. In this paper, our approach is to choose one of the linear scheduling alternatives obtained in [9] and develop a nonlinear scheduling function with smaller and parameterized equitemporal zones. Analyzing the four scheduling functions in [9], we choose to partition the equitemporal zones of the scheduling function shown in Fig. 2 for the following reasons:

- The resulting processor array architecture has the best time complexity among the six obtained architectures.
- Simple feeding of data without any delay registers.
- No communication between PEs. Hence, easier partitioning without any feedback connections.
- Partial outputs are stored in local registers in the PEs. Hence, straightforward accumulation of partial results at each time step.

Planes in Fig. 2 that represent equitemporal zones can be partitioned into smaller zones with parameterized dimensions. Rather than assigning all points on a plane the same time value, only points in the new smaller zones are assigned the

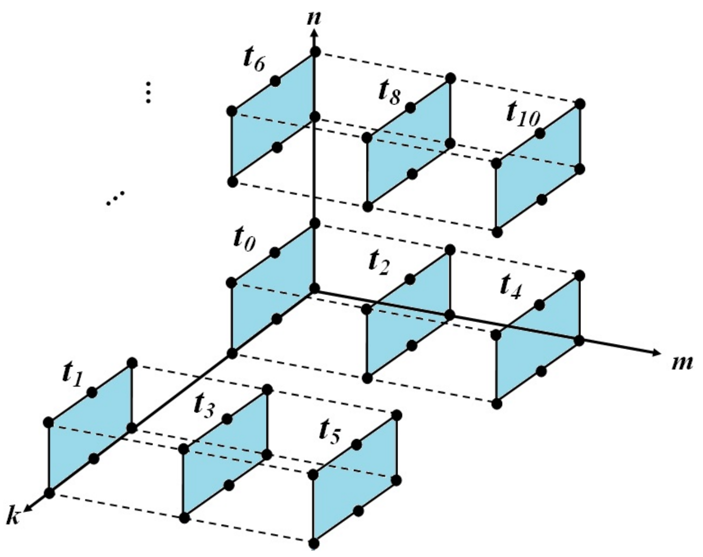

Fig. 3: 2-D equitemporal zones using nonlinear scheduling function (3), K=6, M=3, w_k= 3, and w_n= 2.

same time value. The partitioning adopted in this work along with the order of execution are shown in Fig. 3. Rather than calculating all $N \times K$ distances at each time step using linear scheduling, nonlinear scheduling allows for more control to calculate only $w_k \times w_n$ distances at each time step. The nonlinear scheduling function that assigns time values to points in the computation domain is given by:

$$t(\mathbf{p}) = \left\lfloor \frac{k}{w_k} \right\rfloor + H_k m + H_k M \left\lfloor \frac{n}{w_n} \right\rfloor \tag{3}$$

where:

$$H_k = \left\lfloor \frac{K-1}{w_k} \right\rfloor + 1 \tag{4}$$

C. Projection Operation

Linear projection is defined as the mapping of several points in the n-dimensional computation domain $\mathcal{D}$ to a single point in a k-dimensional domain $\tilde{\mathcal{D}}$, where $k \leq n$ [12]. It simply eliminates coordinates of axes along the projection direction vectors with no control on the size of the projected processor array. Previously obtained architectures in the literature are 1-D or 2-D processor arrays of size K, M, or N in each dimension. For more control on the size of the resulting processor array and to achieve full utilization of hardware resources, we choose to assign each point in the equitemporal zones described in the previous subsection to a PE in the projected processor array. This ensures that all PEs are always busy with more control on the size of the resulting architecture by choosing parameters w_k and w_n to meet area and I/O bandwidth constraints. Each point $\mathbf{p} = \begin{bmatrix} k & m & n \end{bmatrix}^t \in \mathcal{D}$ will be mapped to processing element $PE_{i,j}$ where:

$$i = k \bmod w_k \tag{5}$$

$$j = n \bmod w_n \tag{6}$$

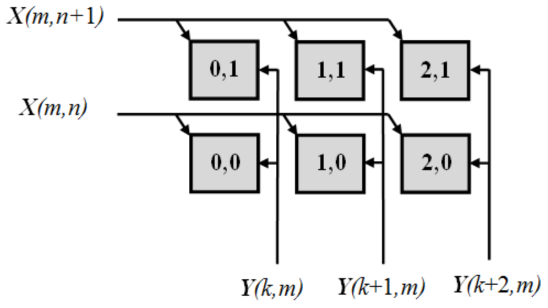

Fig. 4: Proposed processor array architecture with w_k= 3, and w_n= 2.

IV. The proposed processor array architecture

The processor array architecture corresponding to the nonlinear scheduling and projection described in the previous section is shown in Fig. 4. This architecture is obtained by partitioning the original $K \times N$ processor array we proposed in [9] using linear scheduling and projection operations. The proposed architecture is a parameterized processor array of $w_k \times w_n$ PEs. Both inputs $\mathbf{X}$ and $\mathbf{Y}$ are broadcast and output $\mathbf{D}$ is localized. As shown in Fig. 5, partial results of output variable $\mathbf{D}$ calculated at each time step for a given value of m are stored in local registers and hence, simple accumulation of outputs at each time step is possible.

In the case of K-Means clustering algorithm, for instance, a total of $w_k \times w_n$ distances between w_n samples of input data $\mathbf{X}$ and w_k cluster centroids are generated every M clock cycles by the proposed architecture. The total number of cycles required to calculate all elements of distance matrix $\mathbf{D}$ is:

$$C_{total} = \lceil K/w_k \rceil \lceil N/w_n \rceil M \tag{7}$$

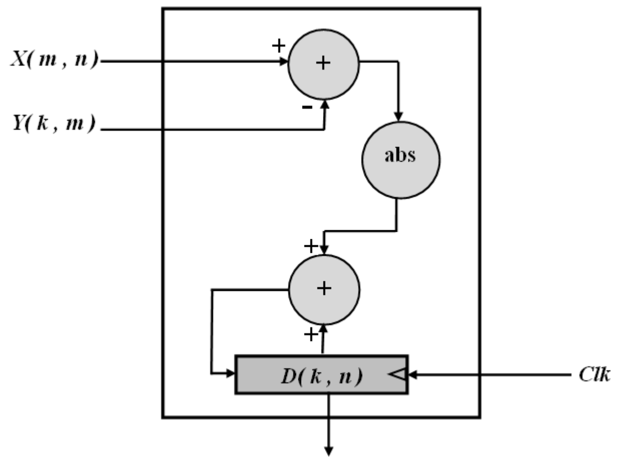

Fig. 5: Processing element structure for the proposed architecture.

V. Design Comparison

The 2-D processor array proposed in [7] has $K \times N$ PEs. For datasets of thousands or even millions of samples N, the proposed architecture is not feasible for hardware implementation as it consists of a huge number of PEs with large number of data inputs being fed simultaneously. To overcome these limitations, the authors of [8] have proposed another 2-D processor array architecture of $K \times M$ PEs. However, the proposed architecture is not practical for high-dimensional datasets with large number of features M per sample as it requires feeding all features of each sample simultaneously. In both architectures, data is fed in a skewed pattern and hence, a large number of delay registers are required for large and high-dimensional datasets.

The linear processor array architecture proposed in [6] requires less area and I/O pins compared to 2-D architectures. However, the time complexity is much higher since lesser number of PEs are involved, with only one feature of each sample of dataset $\mathbf{X}$ being fed at a time.

The proposed architecture scales better for high-dimensional data since it deals only with one dimension of data at each time step. As shown in Table I, the proposed architecture is fully parameterized. Parameters w_k and w_n can be chosen to determine the number of PEs so that area and I/O bandwidth constraints are met.

VI. Implementation Results

The proposed processor array architecture along with the 2-D and 1-D processor arrays previously obtained in [8] and [6], respectively, are implemented on FPGA, to accelerate distance computation involved in clustering data samples of the letter recognition dataset from UCI Repository [13]. The dataset consists of 20,000 samples with each having 16 numerical attributes represented as integer values in the range from 0 to 15. The three architectures are implemented in Verilog hardware description language using Xilinx ISE Design Suite 13.4 to target Xilinx Virtex7 XC7VX330T. Table II and Fig. 6 show implementation results for distance calculation involved in one iteration of the K-Means clustering algorithm with M =16 features, N=20,000 samples, and different number of clusters K. Execution times are calculated using the maximum frequencies achieved after implementing each architecture and the required clock cycles shown in the fourth column of Table I.

Implementation results show that the proposed architecture achieves the best compromise between area and speed. The linear architecture in [6] has the worst Area-Delay product due to delay registers that add up to its area complexity. The proposed architecture achieves an Area-Delay product that is comparable and even slightly better than that of the 2-D architecture in [8], using an average of only 6% and 48% of its area and I/O pins, respectively.

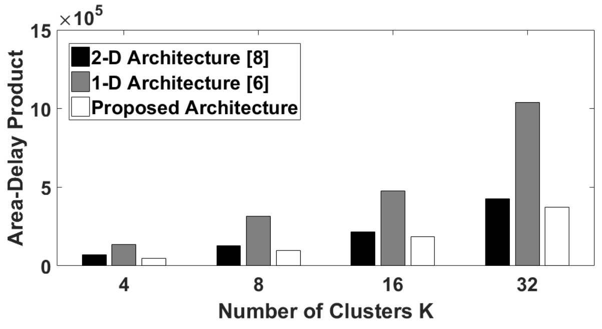

Fig. 6: Area-Delay Product.

VII. Conclusion

A systematic methodology to develop scalable processor array architecture for similarity distance computation is presented in this paper. Compared to previous processor arrays that have been developed using ad hoc techniques, the proposed architecture is realized based on systematic full design space exploration of the distance computation problem.

The fully parameterized architecture proposed achieves better compromise between area and speed with more control on the number of PEs and I/O pins. Moreover, the methodology introduced makes scalability possible in the designed architecture.

References

[1] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[3] T.-M. Huang, V. Kecman, and I. Kopriva, *Kernel based algorithms for mining huge data sets*. Springer, 2006, vol. 1.

[4] S. Kotsiantis, "Supervised machine learning: a review of classification techniques," *Informatica*, vol. 31, no. 3, pp. 249–269, 2007.

[5] A. Choudhary, R. Narayanan, B. Ö. Ikyılmaz, G. Memik, J. Zambreno, and J. Pisharath, "Optimizing data mining workloads using hardware accelerators," in *Proc. of the Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, 2007.

TABLE I: Design Comparison

Design	Scheduling and Projection	Circuit Complexity	Time Complexity	Inputs	Outputs
2-D Processor Array [7]	Linear	$K \times N$ PEs $\frac{1}{2}(K^2 + N^2 - K - N)$ Registers	$K + M + N$	$K + N$	$K \times N$
2-D Processor Array [8]	Linear	$K \times M$ PEs $\frac{1}{2}M(M-1)$ Registers	$K + M + N$	M	K
1-D Processor Array [6]	Linear	K PEs $\frac{1}{2}K(K-1)$ Registers	$K + MN - 1$	$K + 1$	K
Proposed	Nonlinear	$w_k \times w_n$ PEs	$\lceil K/w_k \rceil \lceil N/w_n \rceil M$	$w_k + w_n$	$w_k \times w_n$

TABLE II: Implementation Results

K	**2-D Architecture [8]**			**1-D Architecture [6]**			**Proposed Architecture** $w_k = K/2$, $w_n = 2$		
	Area (# Slices)	**Execution Time (μ Sec)**	**Bonded IOBs**	**Area (# Slices)**	**Execution Time (μ Sec)**	**Bonded IOBs**	**Area (# Slices)**	**Execution Time (μ Sec)**	**Bonded IOBs**
4	664	105	338	76	1799	93	32	1520	88
8	1192	107	370	152	2075	157	60	1612	136
16	1988	108	434	249	1916	285	114	1624	232
32	3880	109	562	488	2130	533	217	1713	424

[6] M. F. Hsieh and C. H. Lai, "A serial input VLSI systolic architecture for a clustering analyser," *International journal of electronics*, vol. 84, no. 3, pp. 269–284, 1998.

[7] H. Cheng and C. Tong, "Clustering analyzer," *Circuits and Systems, IEEE Transactions on*, vol. 38, no. 1, pp. 124–128, 1991.

[8] M. Lai, M. Nakano, Y. Wu, and C. Hsieh, "VLSI design of clustering analyser using systolic arrays," in *Computers and Digital Techniques, IEE Proceedings-*, vol. 142, no. 3. IET, 1995, pp. 185–192.

[9] A. Kanan, F. Gebali, and A. Ibrahim, "Design space exploration of 2-D processor array architectures for similarity distance computation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2218–2228, Aug 2017.

[10] D. G. Perera and K. F. Li, "Parallel computation of similarity measures using an FPGA-based processor array," in *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*. IEEE, 2008, pp. 955–962.

[11] F. Gebali, *Algorithms and Parallel Computing*. John Wiley & Sons, 2011.

[12] F. El-Guibaly and A. Tawfik, "Mapping 3-D IIR digital filter onto systolic arrays," *Multidimensional Systems and Signal Processing*, vol. 7, no. 1, pp. 7–26, 1996.

[13] C. Blake and C. Merz, "UCI repository of machine learning databases." *[Online]. Available: http://www. ics. uci. edu/˜ mlearn/ML-Repository. html*, 1998.