VLSI design of clustering analyser using systolic arrays

M.F. Lai
M. Nakano
Y.P. Wu
C.H. Hsieh

Indexing terms: VLSI design, Clustering analyser, Systolic array

Abstract: This paper presents a VLSI architecture for clustering analysers. The proposed VLSI architecture exploits two-dimensional systolic arrays, which use a high degree of parallel and pipelined processing. The architecture dramatically reduces the immense number of processing elements (PEs) which are required by previous architectures. Moreover, the same architecture can be utilised for applications with a variable number of input patterns. Also, unlike previous architectures, the patterns are applied to the inputs in serial format, which can save a large number of pin counts, and therefore the proposed architecture is very attractive for VLSI implementation. Using the proposed architecture, the complexity of VLSI circuit of the clustering analyser can be reduced significantly.

1 Introduction

Clustering analysis is a useful technique in the field of pattern recognition and image processing. Applications of clustering analysers include image compression, image segmentation, image registration, speech recognition, pattern recognition etc. [1–3]. A number of clustering techniques have been developed [4–6]. The most popular clustering technique for applications in pattern recognition and image processing is the squared error clustering technique. Since the squared error clustering technique involves an enormous amount of computations, the reduction of total computation time is essential for real-time applications. Recently, the progress in VLSI technology has made possible the implementation of the squared error clustering technique by hardware.

Several VLSI architectures based on the squared error clustering technique were presented earlier [7–9]. A VLSI architecture using a two-level pipelined design was proposed [7]. In this architecture, the time complexity for one pass is $O(N \times \max(K, M))$, where N is the total number of input patterns, K is the number of clusters, and M is the dimension of the input pattern vector. Another VLSI architecture with time complexity of $O(N)$ was proposed [9]. The hardware complexity is roughly estimated as $N \times K$ PEs, $(N-1)$ registers and N buffers for module A; $M \times K$ PEs for module B; $N(M+2)$-2 shift registers for module C; N PEs and $(N-1)$ registers for module D. Immense quantities of PEs and registers are required in modules A, C and D since N is often very large in most clustering applications. Moreover, all N patterns are input in parallel; thus VLSI implementation is impractical.

A VLSI architecture using systolic arrays to realise the squared error clustering technique is proposed in this paper. The time complexity of the proposed architecture is found to be $O(N)$. The hardware complexity of the proposed architecture is approximately $M \times K$ PEs for module A, $M \times K$ PEs for module B, $M(\frac{3}{2}M + K + \frac{5}{2})$ for module C, and K PEs for module D. It is observed that the hardware complexity in modules A, C and D is reduced drastically compared to Reference 9 because M and K are much smaller than N. Furthermore, the patterns are input in a serial format in the proposed architecture, which can save a lot of pin counts.

2 Squared error clustering technique

2.1 Notation

N = total number of input patterns for a data set to be partitioned into clusters
M = dimension (features) of input pattern vector
P = $N \times M$ pattern matrix consisting of N vectors with M features each
$P(i)$ = $1 \times M$ vector for the ith input pattern, where $1 \leqslant i \leqslant N$
$P(i, m)$ = element in the matrix $P(N \times M)$. $P(i, m)$ represents the mth feature of the ith input pattern, where $1 \leqslant i \leqslant N$, $1 \leqslant m \leqslant M$
K = total number of clusters
C = $K \times M$ cluster centre matrix consisting of K vectors with M features each
$C(k)$ = $1 \times M$ vector for the kth cluster centre, where $1 \leqslant k \leqslant K$
$C(k, m)$ = element in the matrix $C(K \times M)$. $C(k, m)$ represents the mth feature of the kth cluster centre, where $1 \leqslant k \leqslant K$, $1 \leqslant m \leqslant M$
$L(i)$ = label assigned to the ith pattern $P(i)$. $L(i)$ indicates the cluster number which the ith pattern belongs to, where $1 \leqslant L(i) \leqslant K$, $1 \leqslant i \leqslant N$

Paper 1790E (C2), first received 5th April and in revised form 9th December 1994

M.F. Lai and Y.P. Wu are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, Republic of China
M. Nakano is with the Department of Control Engineering, Tokyo Institute of Technology, Tokyo 152, Japan
C.H. Hsieh is with the Department of Electrical Engineering, Chung Cheng Institute of Technology, Tashi, Taoyuan, Taiwan, Republic of China

$S(k)$ = set of input patterns belonging to the cluster k

$$S(k) = \{i \mid L(i) = k, \quad \text{for } 1 \leqslant i \leqslant N\},$$

$$\text{where } 1 \leqslant k \leqslant K \quad (1)$$

$|S(k)|$ = total number of input patterns belonging to the cluster k, where $1 \leqslant k \leqslant K$

D_i^k = the squared distance between the ith input pattern and the kth cluster centre, which is defined as

$$D_i^k = d^2[P(i), C(k)] = \sum_{m=1}^{M} [P(i, m) - C(k, m)]^2 \quad (2)$$

$min\ (D_i)$ = the minimum distance for the ith input pattern. min (D_i) = is the smallest value of the squared distances between the ith pattern and all cluster centres computed from eqn. 2

$E^2(k)$ = the squared error for the kth cluster, which is defined as

$$E^2(k) = \sum_{i \in S(k)} \min\ (D_i), \quad \text{where } 1 \leqslant i \leqslant N \quad (3)$$

2.2 Squared error clustering algorithm

A number of methods have been developed for the realisation of the squared error clustering technique, such as the K-means method, ISODATA method, Forgy's method, and dynamic clustering algorithm, etc. [10–12]. These methods employ the same main algorithm, and the only difference is the heuristic strategy. This paper proposes a systolic array architecture for the main algorithm of the squared error clustering technique.

The major tasks of the squared error clustering algorithm include the labelling process and cluster centre updating process. The labelling process is to assign a suitable label $L(i)$ for each input pattern according to the following criterion:

$$L(i) = k \quad \text{if } D_i^k < D_i^j, \quad \text{for } k \neq j,$$

$$\text{where } 1 \leqslant k \leqslant K, 1 \leqslant j \leqslant K \quad (4)$$

The labelling process continues until the last label $L(N)$ is generated.

The cluster centre updating process is used to update the centre co-ordinates of each cluster if the labelling process has changed the labels for input patterns. Two methods can be used to determine when the cluster centres will be updated [7]. In the first method, the updating is performed when each label is generated during the labelling process. In the second method, the cluster centres remain unchanged during the labelling process. The update of new cluster centres will be enabled only after the completion of the labelling process on the entire data set. The latter method is used in this paper because it is more suitable for systolic array architecture. After the last label $L(N)$ is generated, the new cluster centres are calculated according to the following formula:

$$C(k, m) = \frac{1}{|S(k)|} \sum_{i \in S(k)} P(i, m),$$

$$\text{where } 1 \leqslant k \leqslant K, 1 \leqslant m \leqslant M \quad (5)$$

The labelling process and cluster centre updating process are performed repeatedly until the convergence condition is met.

3 System operation and organisation

Due to the iterative nature of the squared error clustering technique, huge numbers of arithmetic operations (including addition, subtraction, squaring, and magnitude comparison) are required even for ordinary number of input patterns. This demands a considerable amount of CPU time to perform the squared error clustering algorithm [7]. Though the clustering analysis for many applications can be performed by offline, there exist some applications where online processing is required. For example, in the application of video compression using frame-adaptive vector quantisation [13, 14], the new codebook of the current frame must be generated by using clustering analysis to update the previous codebook. Hence, high speed online clustering analysis results are needed for real-time operation. VLSI implementation of the squared error clustering technique is very favourable for the high speed applications [15, 16]. A systolic array using a high degree of parallel processing and pipelining is helpful to speed up the operation further [17, 18].

Fig. 1 illustrates system organisation of the proposed VLSI architecture for the online clustering analysis. The architecture is composed of four modules (modules A, B, C and D). Module C arranges data flow in an appropriate timing sequence for module A and module B. As depicted in Fig. 1, input pattern $P(1, 1)$ is applied to module A at the first clock; $P(2, 1)$ and $P(1, 2)$ are applied to module A at the second clock; $P(3, 1)$, $P(2, 2)$ and $P(1, 3)$ are applied to module A at the third clock, etc.

Module A obtains skewed input patterns from module C. It computes the squared distance D_i^k between every input pattern vector $P(i)$ and each cluster centre vector $C(k)$ to generate the minimum squared distance min (D_i) and the corresponding label $L(i)$ for each input pattern sequentially at every clock cycle. $L(i)$ is applied to module B and module D for decoding, while min (D_i) is applied to module D for calculation of the squared error $E^2(k)$ for the kth cluster.

Module B obtains data stream from module C in a vector-parallel scheme. The ith pattern vector $P(i)$ is classified into the corresponding cluster designated by $L(i)$ for the computation of the cluster centres indicated in eqn. 5. After the calculation of the new cluster centres is completed, the new cluster centres are transferred from module B to the corresponding registers in module A.

Module D receives min (D_i) and $L(i)$ for each input pattern and computes the squared error $E^2(k)$ for each cluster according to eqn. 3. Module D is employed to check the convergence condition to terminate the clustering process.

4 Module A architecture

The architecture of module A is illustrated in Fig. 2*a*. The PEs are arranged in a two-dimensional systolic array so that parallel processing can be performed for the PEs located on the same oblique dashed line. In addition, pipelined processing is performed for the PEs on the same horizontal row and the C cells in the vertical column.

The input pattern vectors, arranged in a skewed scheme, are applied to PE(1, 1) ~ PE(1, M) in the top row. The input pattern $P(1, 1)$ is applied to PE(1, 1) at the first clock. $P(1, 1)$, after delay of one clock cycle, is passed to PE(2, 1). At this time instant, $P(1, 2)$ is also applied to PE(1, 2). At the third clock, $P(1, 1)$, $P(1, 2)$, $P(1, 3)$ are passed to PE(3, 1), PE(2, 2), and PE(1, 3), respectively.

The internal structure of each PE of module A is shown in Fig. 2*b*. The initial value of each cluster centre element needs to be loaded into the centre register in

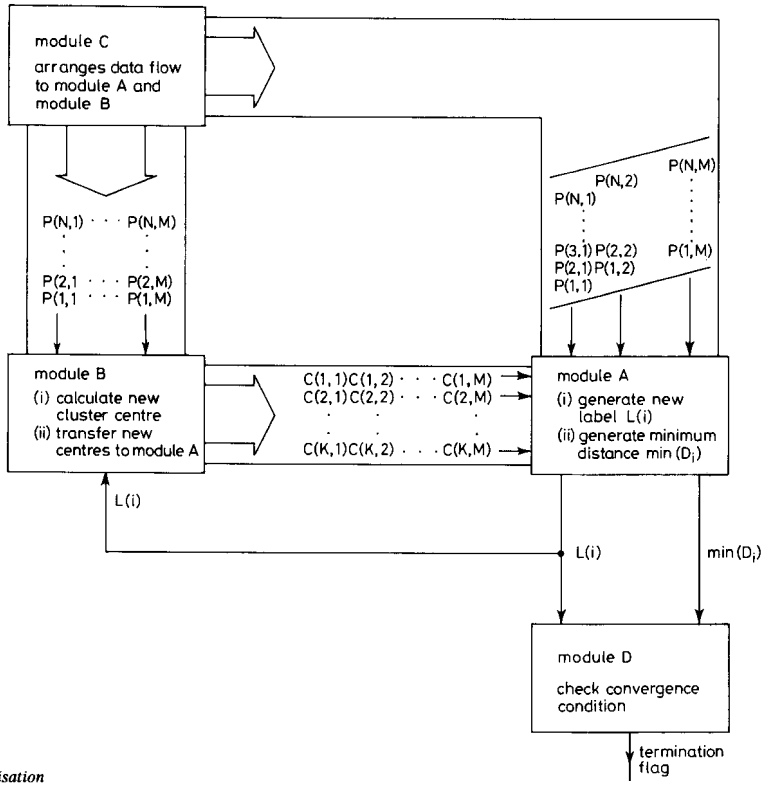
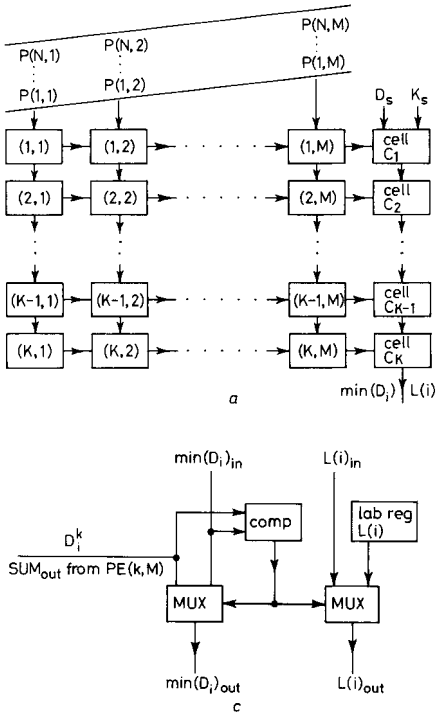


Fig. 1 *System organisation*



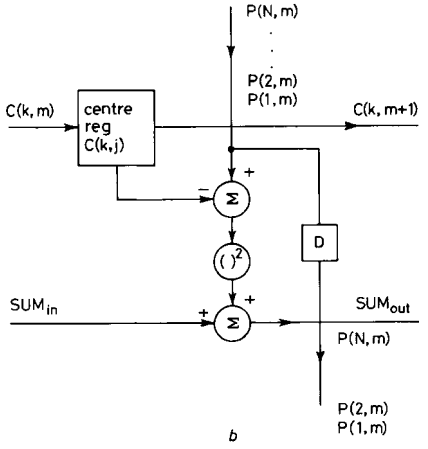


Fig. 2 *Module A architecture*

a Organisation
b Internal structure of PE
c Internal structure of cell *C*

each PE before the clustering operation. The PE subtracts the $P(i, m)$ from $C(k, m)$, squares the difference, and adds the result to the partially computed squared distance (sum_{in}) passed from the left PE. The adder output (sum_{out}) is passed to the right PE for further computation. The input patterns, after delay of one clock cycle through the delay register (D), are passed to the lower PEs. The input path and the output path of the centre register are used for new centre movement from module B to module A during cluster centre updating.

The internal structure of cell C is depicted in Fig. 2*c*. A comparison operation is performed on the two inputs, D_i^k from PE(k, M) and min $(D_i)_{in}$ from upper C cell, and the comparator output is used as a select line for the two multiplexers (muxs). The smaller input and its corresponding label will be output to C cell in the down neighbourhood. The inputs of cell C_1 in the top row are connected to D_s (maximum distortion value) and K_s (dummy index). The output of cell C_K in the bottom row will generate the minimum distance min (D_i) and its associated label $L(i)$.

A simplified two-dimensional array for four clusters ($K = 4$) and three features ($M = 3$) is given in Fig. 3 to

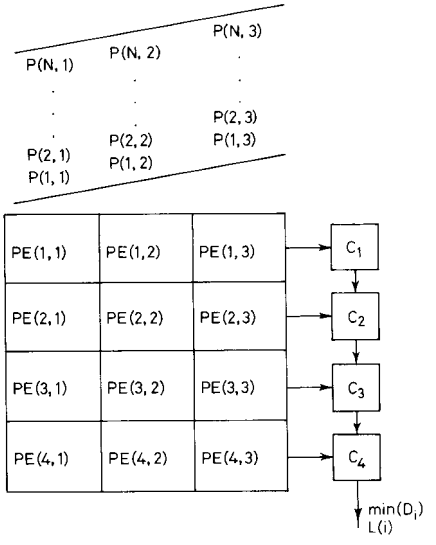


Fig. 3 *Complete systolic array of module A for K = 4 and M = 3*

further illustrate the operation of module A. There are 12 PEs ($K \times M = 12$), and four C cells ($K = 4$). Fig. 4 illustrates the snapshots of the data flow, and the comparison results at each clock cycle for the architecture in Fig. 3. Assume that the first pattern element $P(1, 1)$ is applied to PE(1, 1) at the first clock. Three clock cycles later (i.e. at the fourth clock) $[P(1, 1) - C(1, 1)]^2$ will appear at the output of PE(1, 1) because the PE is constructed by a three-stage pipeline as shown in Fig. 3*b*. Due to the pipelined arrangement of the PEs in the horizontal direction, the partial sum of $\sum_{m=1}^{2} [P(1, m) - C(1, m)]^2$ and $\sum_{m=1}^{3} [P(1, m) - C(1, m)]^2 (= D_1^1)$ will appear at the output of PE(1, 2) and PE(1, 3) at the fifth and sixth clock, respectively. The squared distance D_1^1 is applied to the input of cell C_1 at the sixth clock, and D_1^2, D_1^3, D_1^4 appears at the C_2, C_3, C_4 inputs at the seventh, eighth, ninth clock, respectively. Assume that the operation in C cell takes one clock cycle, then D_1^1 appears at the output of cell C_1 at the seventh clock. Since C cells are arranged in a pipeline in the vertical direction, $\min_{k=1,2} D_1^k$, $\min_{k=1,2,3} D_1^k$, and $\min_{k=1,2,3,4} D_1^k$ will appear at the output of cell C_2, C_3, and C_4 at the eighth, ninth, and tenth clock, consecutively. The minimum distance of the first pattern is generated at the tenth $(1 + M + K + 2 = 10)$ clock, and that of the second pattern at the 11th $(2 + M + K + 2 = 11)$ clock, etc. In conclusion, the label $L(i)$ and the minimum distance min (D_i) for the ith pattern $P(i)$ are generated at the instant of the $(i + M + K + 2)$th clock, and the last label $L(N)$ and the corresponding minimum distance min (D_N) for the last pattern $P(N)$ are generated at the $(N + M + K + 2)$th clock.

5 Module B architecture

The organisation of module B is illustrated in Fig. 5*a*. The functions of PEs in module B include the calculation of new cluster centres and the transfer of cluster centres to module A. The operation of module B starts at the $(M + K + 3)$th clock, when the first new label $L(1)$ is generated from module A. At this instant, the first pattern vector $P(1)$ should also be applied to the module B input. Assume the new label for $L(1)$ is k, then $L(1)$ is decoded by the decoder to select all PE(k, 1) ~ PE(k, M) corresponding to the kth cluster, and the pattern vector $P(1)$ is applied to the kth cluster. This process repeats for all N patterns. The decoder output also increments the counter, which keeps a record of the total number of input patterns belonging to this cluster so far.

The internal structure of PE is shown in Fig. 5*b*. After $L(N)$ is generated at the $(N + M + K + 2)$th clock, all pattern vectors $P(1) \cdots P(N)$ have been classified and accumulated to R_1 registers in the PE of the corresponding cluster. The subsequent step is to calculate new centres by dividing the accumulated values in R_1 registers by the count CT_k ($CT_k = |S(k)|$) in the counter based on the formula indicated in eqn. 5, all PEs can execute the divisions in parallel. Assume the dividers take n clock cycles to complete the divisions for all clusters. The new cluster centres will appear in all R_1 registers at the $(N + M + K + n + 2)$th clock. These new cluster centres should be moved to the centre registers in module A. The cluster centre movement is performed by the multiplexer and register R_2. When $E = 0$, the data in R_1 will be transferred to R_2 at the $(N + M + K + n + 3)$th clock. The next task is to shift the data in R_2 of PE(k, m) in the module B to the PE(k, m) of module A. These data can be moved from module B to the corresponding centre registers in module A in M clock cycles. From the above calculation, a complete one pass operation in module B requires $(N + 2M + K + n + 3)$ clocks.

6 Module C architecture

The organisation of module C is illustrated in Fig. 6*a*. Module C is connected to the memory, which is built with M multiple banks and an address generator. Once presented with a memory address from an address generator, each bank sends out an element of an input pattern. The M elements from all banks form a pattern vector. The delay registers $D, 2D, \ldots, (M - 2)D, (M - 1)D$ in Fig. 6*a*, total number equals $(M/2)(M - 1)$, are utilised to generate the skew format for input patterns to module A.

Module C should also send the data stream of input patterns to module B starting from the $(M + K + 3)$th clock. The serial-in serial-out (SISO) registers in Fig. 6*a*,

total number equals M, are employed to generate a delay of $(M + K + 3)$ clocks for input patterns. Each SISO register comprises $(M + K + 3)$ delay registers as shown in Fig. 6*b*; thus the entire number of registers used in SISO registers is $M(M + K + 3)$. Hence, the total number of registers in module C can be found as

$$M(M + K + 3) + \frac{M}{2}(M - 1) = M(\tfrac{3}{2}M + K + \tfrac{5}{2}).$$

Since $(N + 2M + K + n + 3)$ clock cycles are required for one pass operation in module B, module C should arrange data streams to module A and module B at the period of $(N + 2M + K + n + 3)$ clocks.

7 Module *D* architecture

The organisation of module D is illustrated in Fig. 7*a*. There are K PEs and one decoder. Each PE is used to calculate the squared error of the cluster k according to eqn. 3 and to check whether the condition of $E^2(k) \leqslant \text{THD}$ is met, where THD is a threshold. The label $L(i)$ is applied to the decoder of module D to select one PE from the linear array in Fig. 7*a*. When the PE corresponding to $L(i)$ is selected, the min (D_i) from module A is applied to the selected PE, so that the partial sum of the squared error for the kth cluster $(L(i) = k)$ is computed.

The internal structure of each PE in module D is shown in Fig. 7*b*. The adder and register R_1 are used to accumulate the minimum distances min (D_i) from Module A. When $L(N)$ is generated from module A at the $(N + M + K + 2)$th clock, the R_1 for the kth cluster will contain the squared error $E^2(k)$. The subsequent step is to compare the content of register R_1 with the preloaded threshold (THD) stored in the register R_2. All PEs can perform the comparison simultaneously, and thus it takes one clock cycle to obtain the comparison results in all PEs. When the F_{out} signals of all PEs are true, the F_{out} from the bottom PE is true, and the termination flag is generated. This implies that the squared errors $E^2(k)$ for all clusters meet the convergence condition. Assume that it takes one clock cycle to generate the termination flag when F_{out} signals from all PEs are true, then a complete one pass operation in module D requires $(N + M + K + 4)$ clock cycles.

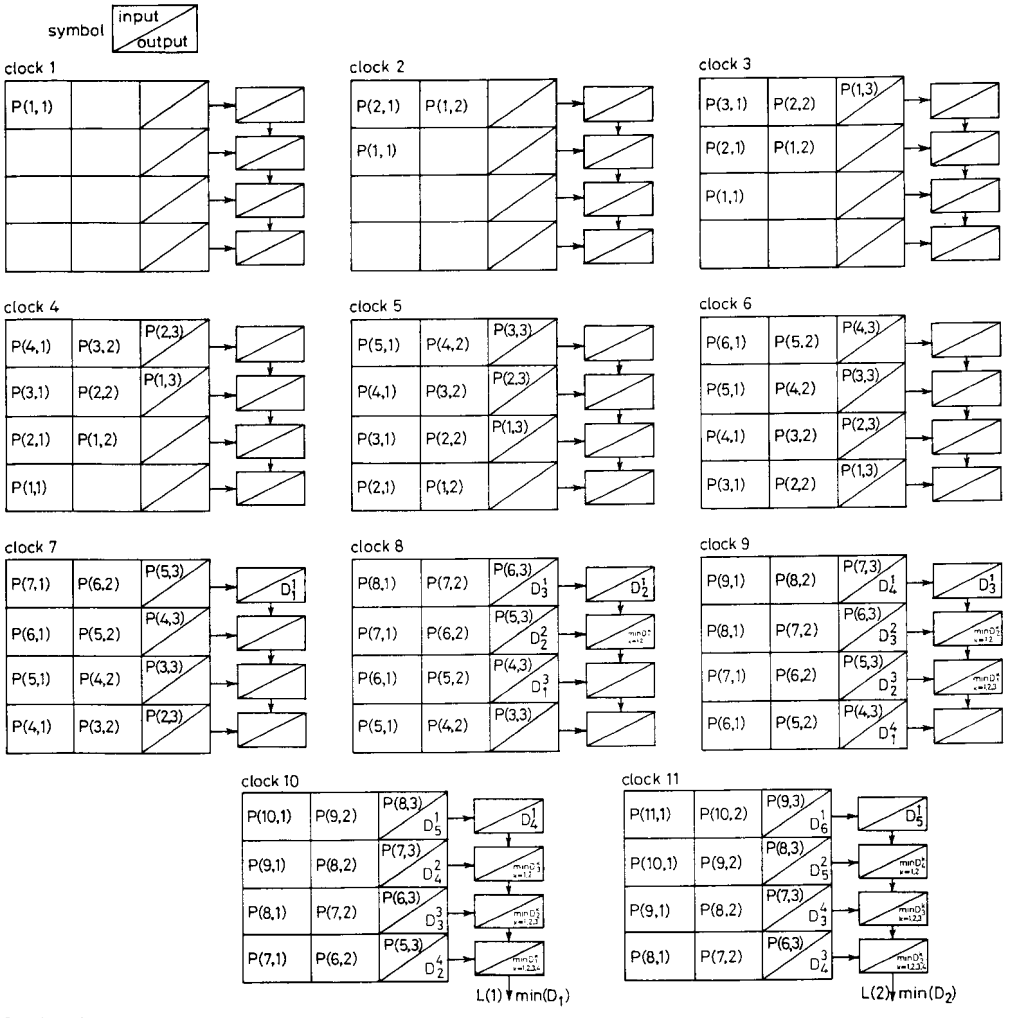


Fig. 4 *Snapshots of data flow in each clock for architecture in Fig. 3*

8 Results and performance evaluation

In this Section, the circuit and computational complexities of the proposed architecture are evaluated. The results are outlined by a comparison against the architecture presented in Reference 9, which are summarised in Table 1.

Table 1: Summary of circuit complexity

	Proposed architecture	Architecture in Reference 9
module *A*	$M \times K$ PEs	$N \times K$ PEs $(N-1)$ registers N buffers
module *B*	$M \times K$ PEs 1 decoder K counters	$M \times K$ PEs $(K-1)$ registers 1 decoder
module *C*	$M(\frac{3}{2}M + K + \frac{5}{2})$ registers	$N(M+2) - 2$ registers
module *D*	K PEs 1 decoder	N PEs $(N-1)$ registers N 2-input AND gates

Since N is normally much greater than M and K in clustering analysis, in contrast to Reference 9 the proposed architecture can save huge amount of hardware for modules *A*, *C* and *D*. For a typical example of a codebook design for vector quantisation, if $M = 16$ and codebook size $K = 256$ with 4 frames of 512×512 images employed, $N = (4 \times 512 \times 512/16) = 65\,536$ will be required. For module *A*, only 4096 PEs are used in the proposed architecture in comparison with 16.7 million $(=2^{24})$ PEs required by the architecture in Reference 9. For module *C*, only 4520 registers are used in our architecture, while 1 179 646 registers are demanded in Reference 9. For module *D*, only 256 PEs are employed instead of the 65 536 PEs used in Reference 9.

The execution time for one pass operation of the proposed architecture is found to be $N + 2M + K + n + 3$ clock cycles in comparison to the $N + M + K + 3$ time units in Reference 9, where only one clock cycle is assumed to perform the division. Since N is much greater than M, K and n, the time complexity of both architectures is $O(N)$.

All N patterns are input in parallel for the architecture in Reference 9, and thus the input bandwidth is N. On the contrary, since the patterns are input in a vector parallel pattern-serial format in our architecture, the required input bandwidth is M. It is obvious that the input bandwidth of our architecture is significantly reduced. Thus, the proposed architecture saves a lot of pin counts, thereby it is more attractive for VLSI implementation.

9 Conclusions

A novel VLSI architecture based on systolic arrays for the squared error clustering technique has been proposed. The architecture is modular, highly regular and

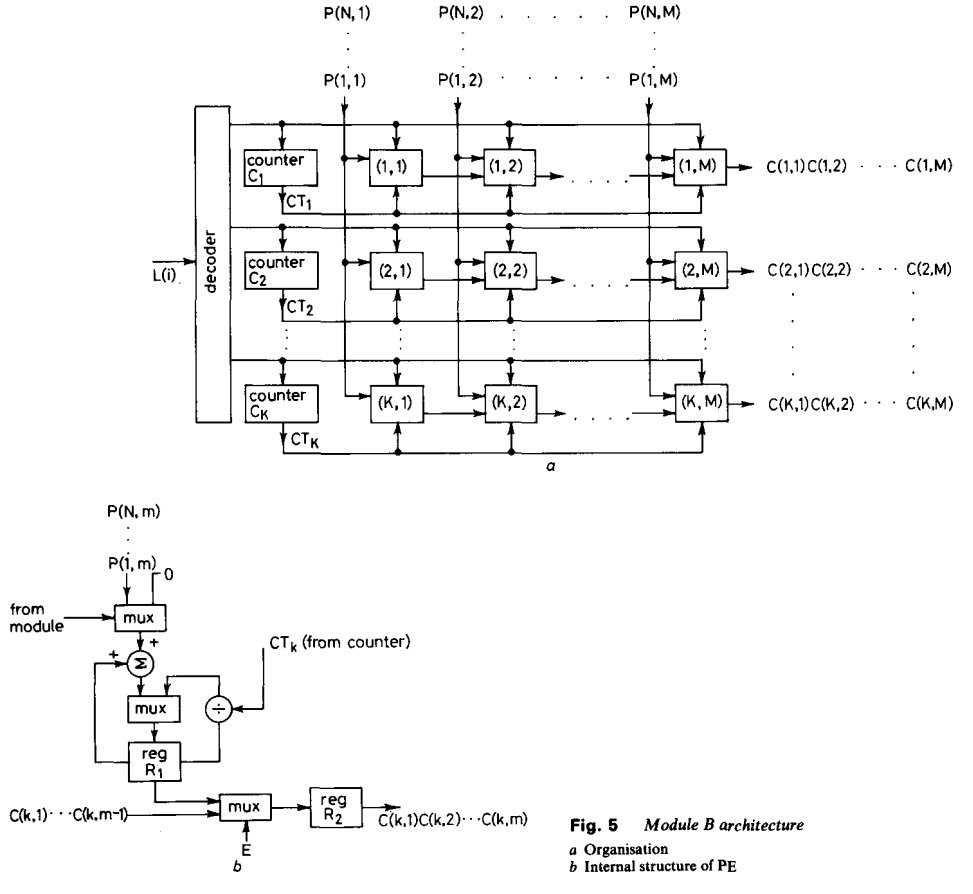


Fig. 5 *Module B architecture*
a Organisation
b Internal structure of PE

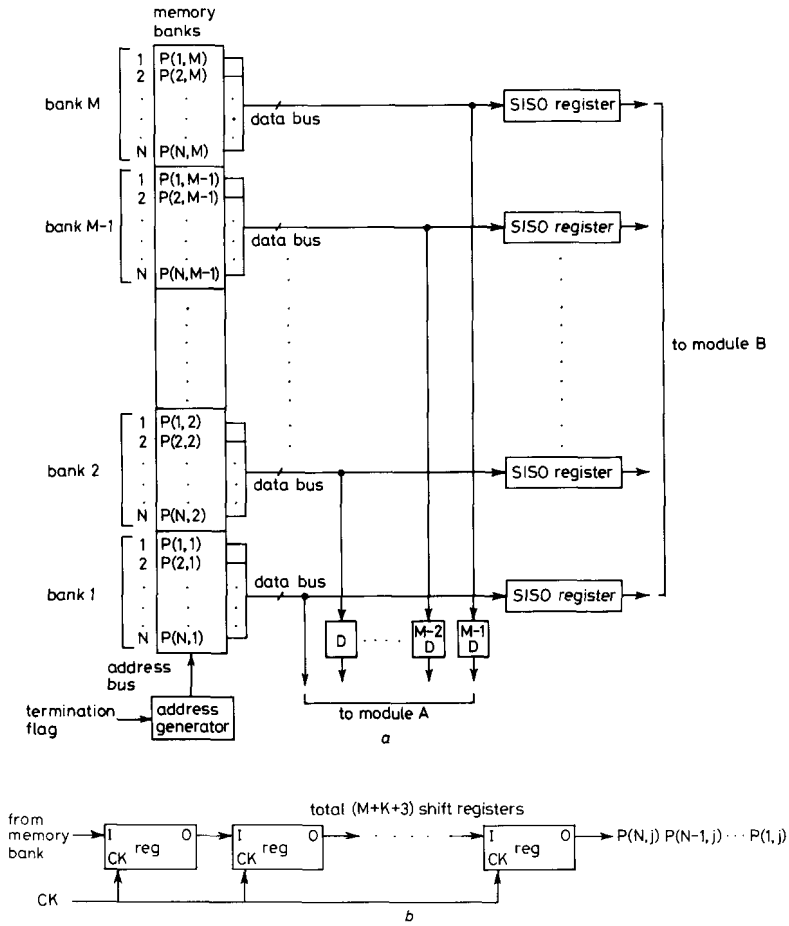


Fig. 6 *Module C architecture*
a Organisation
b Internal structure of SISO shift register

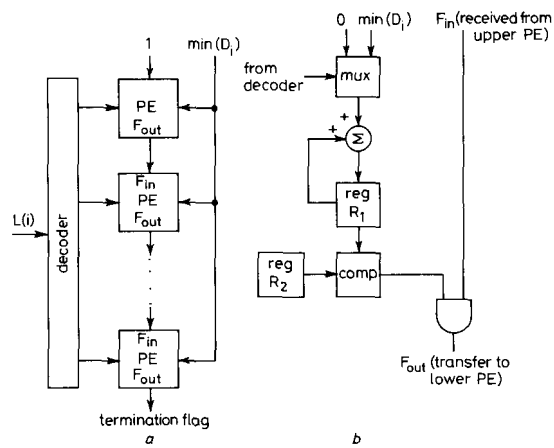


Fig. 7 *Module D architecture*
a Organisation
b Internal structure of PE

with low I/O bandwidth, and thus is very suitable for VLSI implementation. Furthermore, the hardware complexity of the proposed architecture has been reduced dramatically. Based on the proposed architecture, the VLSI design for high speed clustering application can be realised cost effectively.

In addition to the squared distance metric, there are several metrics to measure the similarity, such as absolute distance metric, and weighted squared error metric, etc. [11]. The idea and architecture of this paper can easily be extended to those different metrics. For example, only the squaring circuit in the PE of module *A* in the proposed architecture needs to be modified for absolute distance metric operation.

10 References

1 ANDERBERG, M.R.: 'Cluster analysis for applications' (Academic Press, 1973)
2 DUBES, R.C., and JAIN, A.K.: 'Clustering methodologies in exploratory data analysis', *Adv. Computers*, 1980, **19**, pp. 113–128
3 COLEMAN, G.B., and ANDREWS, H.C.: 'Image segmentation by clustering', *Proc. IEEE*, 1979, **67**, pp. 773–785

4 JAIN, A.K., and DUBES, R.C.: 'Algorithms for clustering data' (Prentice-Hall, 1988)
5 EQUITZ, W.H.: 'A new vector quantization clustering algorithm', *IEEE Trans.*, 1989, **ASSP-37,** (10), pp. 1568–1575
6 DUBES, R.C., and JAIN, A.K.: 'Clustering techniques: the user's dilemma', *Patt. Recog.*, 1976, **8,** pp. 247–260
7 NI, L.M., and JAIN, A.K.: 'A VLSI systolic architecture for pattern clustering', *IEEE Trans.*, 1985, **PAMI-7,** pp. 80–89
8 CHENG, H.D., and TONG, C.: 'Clustering analysis for pattern recognition'. Proceedings of SPIE visual communication and image processing, Cambridge, MA, 1989
9 CHENG, H.D., and TONG, C.: 'Clustering analyser', *IEEE Trans.*, 1991, **CS-38,** (1), pp. 124–128
10 TOU, J.T., and GONZALEZ, R.C.: 'Pattern recognition principles' (Addison-Wesley, 1974)
11 GERSHO, A., and GRAY, R.M.: 'Vector quantization and signal compression' (Kluwer Academic, 1992)
12 KITTLER, J., and PAIRMAN, D.: 'The optimality of reassignment rules in dynamic clustering', *Patt. Recog.*, 1988, **21,** (2), pp. 169–174
13 GOLDBERG, M., and SUN, H.F.: 'Image sequence coding using vector quantization', *IEEE Trans.*, 1986, **COM-34,** pp. 703–710
14 GOLDBERG, M., and SUN, H.F.: 'Frame adaptive vector quantization for image sequence coding', *IEEE Trans.*, 1988, **COM-36,** pp. 629–635
15 KUNG, S.Y.: 'VLSI array processors' (Prentice-Hall, New Jersey, 1988)
16 HSIEH, C.H., and LIN, T.P.: 'VLSI architecture for block-matching motion estimation algorithm', *IEEE Trans.*, 1992, **CSVT-2,** (2), pp. 169–175
17 KUNG, H.T.: 'Why systolic architectures?', *Computer*, 1982, **15,** (1), pp. 37–45
18 DAVIDSON, G.A., CAPPELLO, P.R., and GERSHO, A.: 'Systolic architecture for vector quantization', *IEEE Trans.*, 1988, **ASSP-36,** (10), pp. 1651–1664