Low Cost Architecture for Structure Measure Distance Computation

J.Aranda[1], J.Climent[1], A.Grau[1], A.Sanfeliu[2]
[1] Automatic Control and Computer Engineering Department.
[2] Institut de Robòtica i Informàtica Industrial.
Universitat Politecnica de Catalunya (UPC).

Abstract

Huge and expensive computation resources are usually required to perform graph labelling at high speed. This fact restricts an extensive use of this methodology in industrial applications such as visual inspection. A new systolic architecture is presented which computes structural distances between cliques of different graphs based on a modified incremental Levenshtein distance algorithm. The distances obtained are used as a support function for graph labelling using probabilistic relaxation techniques. The proposed architecture computes the distances between k input cliques of an input graph and one reference clique of a reference graph. It does not limit the number of cliques nor cliques complexity of the input graph, so any input graph can be labelled. A low cost solution has been implemented based on FPGAs.

1. Introduction

The graph matching problem can be accomplished by optimising an energy function based on an heuristic defined support function. A new support function has been presented in [7] which uses the Levenshtein distance as a structure measure distance between two sequences of node neighbours. It depends on the current joint probabilities of the external nodes of both cliques, so it has to be computed in every iteration of the relaxation process.

A new algorithm has been proposed in [3] in order to speed up the computation of the dynamic distance between cliques. It uses the encoding scheme proposed by [6]. For every element in the distance matrix, two incremental costs, instead of one absolute cost, are computed. One cost is the difference between the matrix element and its top neighbour, which will be named *incremental vertical cost (ivc)*. The other one is the difference between the matrix element and its left neighbour, and will be named *incremental horizontal cost (ihc)*. This algorithm constructs then two different matrices, one for the incremental vertical costs and another one for the incremental horizontal costs. The values of the elements of these matrices for a given row *i* and a given column *j*, are determined by the expressions:

$MIN[i,j]=min\{min\{ihc[i,j-1],ivc[i-1,j]\}+C,Sub_{\alpha,\beta}[i,j]\}$

$ivc[i,j] = MIN[i,j] - ihc[i-1,j]$

$ihc[i,j] = MIN[i,j] - ivc[i,j-1]$

where C is the programmable insertion and deletion costs, which have been considered equal for hardware simplicity. $ihc[i,0] = C$ and $ivc[0,j] = C$. $Sub_{\alpha,\beta}$ is the matrix of substitution costs between sequences α and β.

The distance between two non-cyclic sequences is determined then by both the formulae:

$$d(\alpha,\beta) = m \cdot C + \sum_{i=1}^{i=n} ivc[i,m] = n \cdot C + \sum_{j=1}^{j=m} ihc[n,j]$$

Let C_F^λ be a reference clique with a central node v^λ and m external nodes represented by the node sequence V_F^λ. Let C_A^γ be an input clique with a central node v^γ and n external nodes represented by the node sequence V_A^γ. The reference sequence, V_F^λ, is a cyclic sequence. Hence, the dynamic distance d_d between an input clique C_A^γ and a reference clique C_F^λ is determined by the expression:

$$d_d = min\left\{d\left(V_A^\gamma, (V_F^\lambda)^r\right), r = 0..m-1\right\}$$

where $(V_F^\lambda)^r$ is the *r-th* rotation of the reference sequence.

2. Proposed architecture

Related architectures can be found in [4][5] and [2] which perform Levenshtein distance between strings. These solutions do not treat the problem of programmable external costs nor the matching of cyclic sequences. They also restrict the maximum length of input strings. The proposed architecture overcomes all these drawbacks.

The block diagram of a single processing element is presented in fig. 1. At the same time the incremental costs, *ihc* and *ivc*, are input to the processing element, the corresponding substitution cost is also input. The processing element calculates new incremental costs based on the results of the algorithm presented in the introduction. The incremental vertical cost is then transmitted to the adjacent processor to the right, while the incremental horizontal cost is transmitted down. It can

be seen that each element of the edit distance matrix depends only on elements above it and to its left. Then, all elements along a 45° diagonal can be calculated simultaneously.

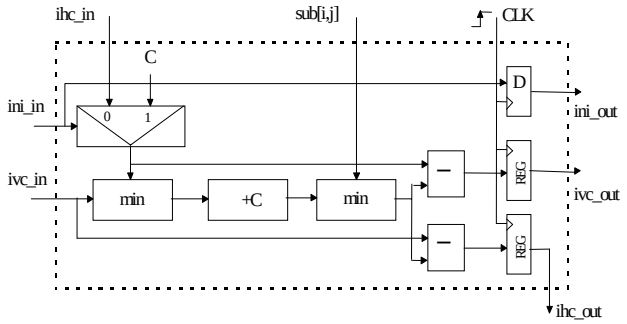

Figure 1. Processing Element

The block diagram of the array architecture to compute the distance between one non-cyclic reference sequence and *k* input sequences is shown in fig. 2. With such a path configuration, each processing element performs computations along its column in the edit distance matrix. The edit distance matrix for comparing a reference sequence of length *m* has *m* such columns. Therefore, *m* processing elements are needed to compute the incremental costs at every cycle. The output of the last column processing element is accumulated in order to compute the edit distance. The width of the accumulator depends on the maximum value that the edit distance can take, and therefore, depends on the edit costs and on the number and length of the input sequences. For this reason, the accumulator is not a part of the architecture and is provided externally.

A single phase is needed to control all data flow. At every new clock cycle, substitution costs corresponding to a new 45° diagonal of the cost matrix are input to the array. At the same time, incremental costs computed by each processing element are latched into the corresponding horizontal and vertical registers and the incremental vertical cost coming from the last processing element is added to the accumulator. The accumulator must be initialised with the value *m*C*.

Once the pipeline array is charged (after the first *m* cycles) a new structure measure distance is computed every n_i cycles (*i=1..k*). As k input reference sequences (with different substitution matrices values) are being compared, an initialisation signal is required every time a new substitution matrix value is input to the first processing element. This signal is propagated by the system clock and it sets the *ihc* initial costs (*C*) and the accumulator initial value (*m*C)*. Fig. 2 shows the state of the system during the *i-th* clock cycle.

For parallel matching of cyclic sequences, *m* processor arrays like the one in fig. 2 have been used. Each array of processing elements, will compute the distance between the input references and a concrete rotation $(V_F^\lambda)^\gamma$ of the reference sequence. This array will be identified with the number of the respective rotation, *r (r=0..m-1)*. Since the reference sequence $(V_F^\lambda)^\gamma$ is obtained just rotating V_F^λ *r* times, the substitution cost matrix $Sub_{V_A^\gamma,(V_F^\lambda)^\gamma}$ will be obtained rotating columns of $Sub_{V_A^\gamma,V_F^\lambda}$ *r* times too:

$$Sub_{V_A^\gamma,(V_F^\lambda)^\gamma}[row,col] = Sub_{V_A^\gamma,V_F^\lambda}[row,((col+r-1)\bmod m)+1]$$

Last expression shows that the substitution cost to be input to the first processing element of each array *r*, during the *i-th* cycle of operation, is the *r+1* column of the *i-th* row of the unrotated substitution cost matrix:

$$Sub_{V_A^\gamma,(V_F^\lambda)^\gamma}[i,1] = Sub_{V_A^\gamma,V_F^\lambda}[i,(r\bmod m)+1] = Sub_{V_A^\gamma,V_F^\lambda}[i,r+1]$$

Fig. 2 also shows that the *j-th* processor in each array *r*, during the *i-th* cycle, needs the substitution cost given by expression:

$$Sub_{V_A^\gamma,(V_F^\lambda)^\gamma}[i-j+1,j] = Sub_{V_A^\gamma,V_F^\lambda}[i-j+1,((j+r-1)\bmod m)+1]$$

which coincides with substitution cost needed by *(j-1)th* processor in array *(r+1)*mod *m*, during the *(i-1)th* cycle:

$$Sub_{V_A^\gamma,(V_F^\lambda)^{\gamma+1}}[(i-1)-(j-1)+1,j-1] = Sub_{V_A^\gamma,(V_F^\lambda)^\gamma}[i-j+1,j]$$

Thus only one row of the substitution matrix have to be input into the proposed architecture at every clock cycle. Substitution costs are then propagated at every new clock cycle through the architecture from processor *j* of array *r* towards processor *j+1* of array *(r-1)*mod *m*.

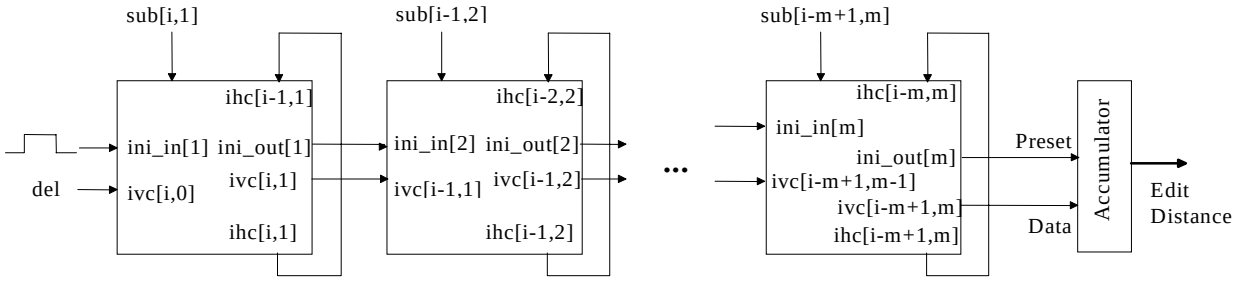

Figure 2. Array of processing elements

3. Architecture implementation and results

Field-programmable gate arrays (FPGA) has been used to implement the proposed architecture. These devices permit to be reprogrammed in just a few milliseconds. This capability permits the *ins* and *del* costs to get a constant but programmable value *C*. Also, processor arrays of variable length can be fitted in the same device.

The major constrain for on-chip implementation are substitution costs, which are inputs to the proposed architecture. A new row of substitution cost matrix is needed by the arrays of processors at every new clock cycle. In order to limit I/O requirements, substitution cost has been limited to the [0..7] rang. Thus only 3 pins are necessary for each array. The only outputs wanted from the arrays are the incremental distance values (4 bits) that have to feed the external accumulators.

For a low cost and fast implementation an ALTERA EPF81500A device from family *Flex 8000* has been chosen. Processing elements and their interconnection has been programmed using AHDL Development Software [1]. This device permits allocation for up to 49 processing elements (7 arrays) what is enough for most of graph labelling applications. This limits the actual implementation to reference cliques of 7 external nodes. For longer reference sequences to be matched a cascadable strategy can easily be performed just appending more than one device before the accumulator. Another solution could be the use of a higher density device of family *Flex 10K* with a capacity for 196 processing elements (14 arrays) and also cascadable.

Time constrains have been analysed in order to evaluate critical paths and minimum clock cycle period. Analysis of critical paths between processing elements has resulted in a maximum delay of 14 nsec for the worst case. However, automatic time analysis tools recommend a maximum clock frequency for the presented implementation of 23.80 MHz. Only one single phase is needed to control the overall process what minimises the use of limited fast interconnect lines.

Max+Plus II Simulator has been used for device functional operation and timing simulation. Fig. 3 shows the incremental distance values (outputs) between two input cliques (V_A^1 and V_A^2) and all the possible rotations of a reference clique V_F^a. External substitution matrices (input) used for this example are also shown in fig. 3. The *C* value has been set to 4 in the exemple.

4. Conclusions

This paper introduces the problem of high speed structure measure distance computation for graph labelling applications. A new systolic architecture is presented which permits to work with externally calculated substitution costs. Only one row of substitution cost matrix is needed by the architecture at every new clock cycle. The proposed architecture computes the distances between *k* input cliques of an input graph and one reference clique of a reference graph. It consists on *m* cascadable array of *m* processing elements, being *m* the number of external nodes of the reference clique. The distance for *k* input cliques with n_i external nodes is computed in $m+\Sigma n_i$ clock cycles ($i=1..k$). Therefore, the architecture does not limit the number of cliques or cliques complexity of the input graph.

References

[1] ALTERA Max+Plus®II. *Programmable Logic Development System. AHDL*. Altera Co. 1995.

[2] H.D. Cheng, and K.S. Fu. "*VLSI Architectures for String Matching and Pattern Matching*". Pattern Recognition, vol.20, n.1, pp. 125-141. 1987.

[3] J.Climent, A.Grau, J.Aranda and A.Sanfeliu. "*Clique-to-Clique Distance Computation Using a Specific Architecture*". Proc. SSPR'98. To be edited.

[4] R.J. Lipton, and D. Lopresti. "*A Systolic Array for Rapid String Comparison*". Chapel Hill Conf. on VLSI, H. Fuchs, ed., Rockville, Md.: Computer Science Press, pp. 363-376. 1985.

[5] D. Lopestri. "*P-NAC: A Systolic array for comparing nucleic acid sequences*". Computer, vol.20, pp. 98-99. 1987.

[6] R.Sastry, N. Ranganathan, and K. Remedios. *"CASM: A VLSI Chip for Approximate String Matching"*. IEEE Trans. Pattern Anal. Mach. Intell. Vol.17, N.8, pp. 824-830. 1995.

[7] F. Serratosa and A. Sanfeliu. "*Function-Described Graphs Applied to 3D Object Representation*". Image Analysis and Processing, 9th ICIAP, Florence, pp. 701-708, 1997.

$$Sub_{V_A^1,V_F^a} = \begin{bmatrix} 4 & 5 & 4 \\ 5 & 5 & 5 \\ 5 & 4 & 5 \end{bmatrix}$$

$$Sub_{V_A^2,V_F^a} = \begin{bmatrix} 4 & 7 & 6 \\ 4 & 7 & 7 \\ 7 & 0 & 3 \\ 6 & 7 & 5 \end{bmatrix}$$

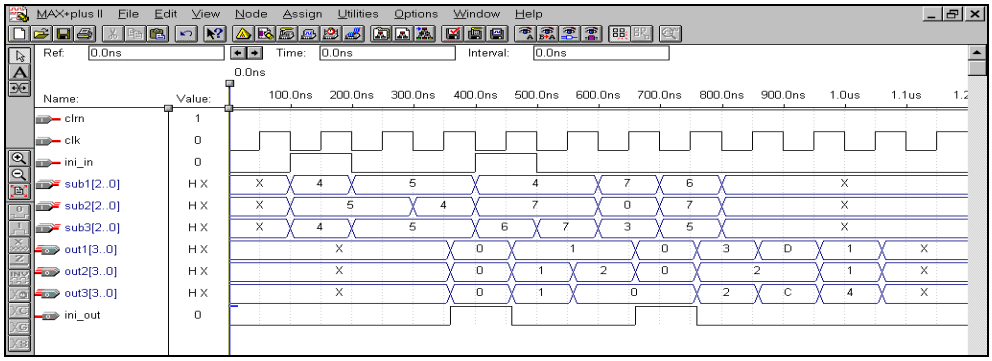

Figure 3. Substitution matrices (inputs) and incremental distance values (outputs) time evolution.