A High Performance Systolic Chip for Spelling Correction

Dominique LAVENIER

IRISA/CNRS, Campus de Beaulieu, 35042 Rennes cedex
FRANCE

Abstract

This paper presents a fully integrated co-processor for accelerating the character string comparison involved in the spelling correction process. The chip we present is based on a truncated 2-D systolic array of 69 processors and is able to perform up to 1.3 Gops. Real time spelling correction is possible on very large vocabularies since dictionaries of 200,000 items can be processed in only 0.1 second.

1 Introduction

The increasing use of computer systems for the processing of textual data has led to a great deal of interest in software for the detection and correction of spelling errors [1]. As a matter of fact, texts are commonly riddled with literals where up to 80 percent of errors are represented by : one letter is wrong ; one letter is missing ; one extra letter is inserted ; or two adjacent characters are transposed.

These errors are respectively referred as substitution, deletion, insertion and transposition errors. They are mainly due to keyboard mistyping during text entry. While detection of erroneous words is an easy task, proposing suitable corrections requires more complex processing.

The correction method we use is based on dynamic programming techniques and provides extremely good results when compared with other techniques. However, it has the major drawback of requiring a large amount of calculation. With dictionaries of few hundred of thousand words the computation time on conventional personal computers, and even on recent workstations, can be unacceptable and inadequate for real-time spelling correction.

A parallel implementation of a comparison algorithm implemented on a systolic arrays can be very efficient. The chip we propose includes a 69 processor array dedicated to spelling correction applications. It can be viewed as an hardware accelerator (co-processor) for comparing one string (erroneous word) with a large set of references belonging to a dictionary. The product of the comparison process is a sequence of words which seem to be suitable for correction.

The paper is organized as follows : section 2 and 3 present respectively the method used for spelling correction (string comparison) and the implementation on a 2-D systolic array. The next section is dedicated to the chip architecture. It is followed by the layout organization. We conclude by indicating performances and work-in-progress.

2 String comparison

The main idea for comparing two strings of characters is to compute a distance between these two strings. This distance, called the *edit distance*, represents the minimal cost for transforming one string into the other by using elementary operations such as substitution, insertion, omission or transposition. Thees operations are called *edit operations*.

Using sequences of such edit operations, any string may be transformed into any other strings. It is then possible to take the smallest number of elementary edit operations required to change one string into another as the measure of the difference between them.

More formally, let $X = (x_1, x_2, ..., x_i, ..., x_n)$ and $Y = (y_1, y_2, ..., y_j, ..., y_m)$ be two strings to compare ; let $d(x, y)$ be the cost of an edit operation to change x into y ; let $\wedge$ represent the null character. It has been shown [2] that the edit distance $D(n, m)$ is obtained from the following recurrence relation :

$$D(i,j) = Min \begin{cases} D(i-1, j-1) + d(x_i, y_j) \\ D(i-1, j) + d(\wedge, y_j) \\ D(i, j-1) + d(x_i, \wedge) \end{cases} \quad (1)$$

The edit operation $d(x_i, \wedge)$ represents the deletion case while $d(\wedge, y_j)$ represents the insertion case. The elementary edit operation $d(x, y)$ may have different values depending on the characters considered.

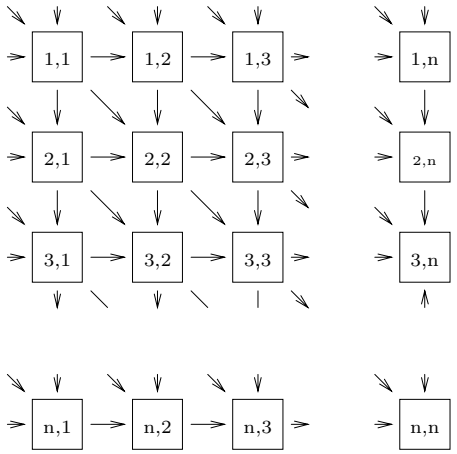

Figure 1: systolic network

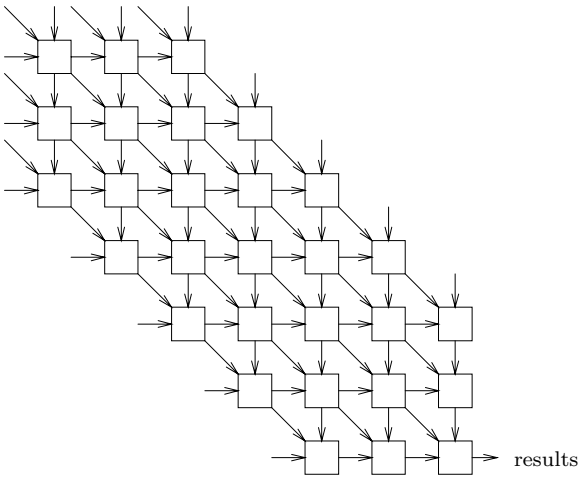

Figure 2: 2-D truncated systolic array

3 Parallel implementation on a systolic array

The systolic implementation we present is based on an assignment of a processing element to each calculation of a value $D(i,j)$. Consider an array of n^2 processors connected as indicated by figure 1. A processor $P(i,j)$ can read data produced by its neighbor processors $P(i-1,j-1)$, $P(i-1,j)$ and $P(i,j-1)$, and propagate its results to processors $P(i+1,j+1)$, $P(i,j+1)$ and $P(i+1,j)$. A systolic cycle represents the elementary distance calculation, that is : the data acquisition ; the minimization operation described by equation (1) ; and the data propagation.

On such a network, parallel execution of a two string comparison requires $2n-1$ systolic cycles, since an elementary distance computation $D(i,j)$ is associated with only one processor $P(i,j)$ and all processors run concurrently.

As a complete comparison lasts $2n-1$ systolic cycles and as a processor is used during only one cycle, a processor, once used, is then available during other cycles for other tasks. Furthermore, the edit distance calculation at time t_k is computed on the diagonal of processors $P(i,j)$ such that $k = i+j-1$. Thus, on each systolic cycle it is possible to start a new comparison.

Pipelining the string comparison on this way allows to increase considerably the efficiency of the network since once the array has been started, a comparison result is produced every systolic cycle.

4 VLSI implementation

4.1 Systolic array dimensions

The average length of words belonging to french or english dictionaries is about 8 characters. The number of words longer than, say, 15 characters is negligible and does not require important computation resources. For these words, the correction process is better left on conventional machines and software.

A 2-D systolic network of 15x15 processors represents a sensible compromise for this specific application. Unfortunately, the resulting 225 processor array remains too large to be integrated on a single chip. However, the number of processors can be drastically reduced as erroneous words are always corrected by words whose length are similar within a range of $\pm$ 2 characters. This means that effective computation is made on a diagonal of the array and that the processors located on the lower left and upper right corners do not contribute actively to the final result.

The systolic array we finally implemented is shown in figure 2. It is a 2-D truncated array with only five diagonals of processors. The median diagonal includes 15 processors ; this gives a total number of processors equal to 69 ($15 + 2 \times 14 + 2 \times 13$).

The figure below gives an idea of the processing element architecture. The **IN** input allows to store, into a dedicated I/O register file, data comming from the other processors (intermediate distance $D(i-1,j-1)$, $D(i,j-1)$ and $D(i-1,j)$). The **CTE** input is connected to a broadcast bus for receiving constant data from the oustside world and from the memory.

These data are stored on a specific constant register before to be used.

As the arithmetic operations involved are very dedicated ones (addition and minimization), two specific units are implemented, namely an adder and a minimizer. These two units are pipelined due to the very regular and repetitive structure of the computation. The accumulator can be loaded either from the adder or from the minimizer. Note that its content can be output on the broadcast bus.

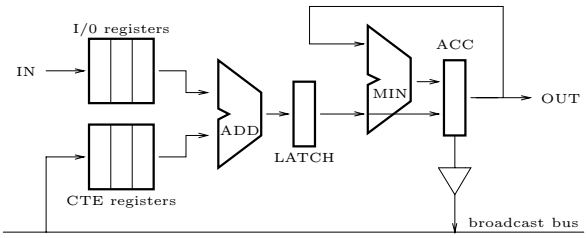

Figure 3: elementary processor

The processors are micro-programmed both for initialization requirement and for implementation of slight variances in string processing algorithms.

4.2 Dataflow and memory management

Remember that one erroneous string must be compared with a large set of references. This means that only data associated with references are flowing (horizontally) through the array while data attached to the erroneous string remains a constant value throughout the correction process. Consequently, in order to perform its local computations, a processor $P(i, j)$ must receive, in each systolic cycle, three intermediate distances from its neighbors and a character x_{ki} corresponding to the i^{th} character of the k^{th} reference.

This flow of data assumes that each processor has a local table for storing substitution cost in order to be able to compute the first term of equation (1). As the characters of the erroneous string do not move during a complete dictionary comparison, the data stored in each processor can be reduced to be only the cost associated with a single character.

Even, this solution consumes considerable area due to the fact that each processor contains its own table. A less expensive approach relies on two observations. The first one is that the tables of all processors in one column are identical. The second is that, assuming that a systolic cycle requires several clock cycles (at least 5), a single memory element can be shared among multiple processors since only one access is needed per processor, per systolic cycle.

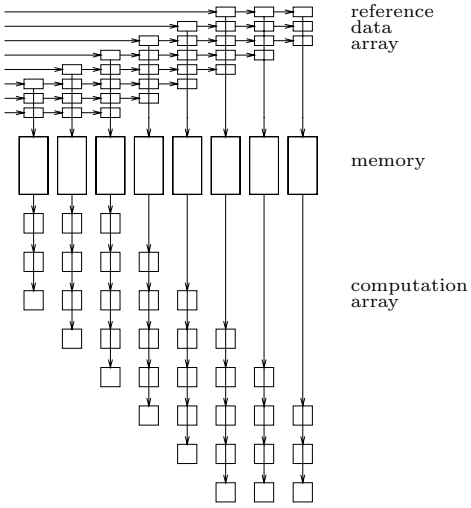

Figure 4: splitting the systolic network into 2 subarrays

Therefore, the systolic array can be divided into 2 subarrays operating in parallel as shown in figure 4. The reference data array is a set of registers containing the reference characters which emulates the reference flow through the systolic network. The registers of one column are used sequentially to address the corresponding memory block and obtain the appropriate substitution cost. The computation array is used to compute the equation (1) in each cell, assuming that the distance cost is available in a register updated by the reference data array. This configuration assumes that the reference data array operates in advance (one systolic cycle ahead).

5 Design methodology

Figure 5 gives the floorplan of the chip. For compactness and suitable VLSI implementation, the systolic array is moved into an orthogonal matrix such that diagonals are laid down horizontally. Furthermore, to get a very regular structure, *dummy* processors are added at the upper left and lower right corners. They are only used during initialization steps and do not participate to the edit distance computation. For the same reasons, the reference data array is also transformed into a regular rectangular matrix.

Each column of the processor array is connected to an independent memory by a broadcast bus. The processors can input data from this bus and store them into registers. References are introduced into

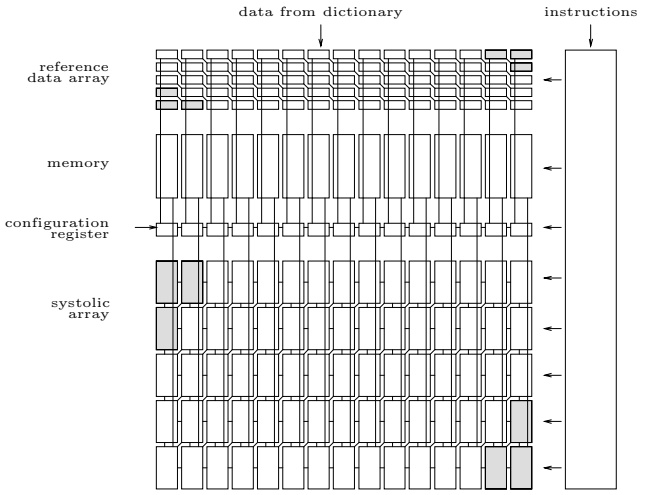

Figure 5: chip organization

the memory by the reference data array.

Before starting the comparison process, some constant registers in the processors must be initialized, as must be the memory. A configuration shift register has been added to perform these different tasks. This shift register is also an effective way to test chip functionality.

The different parts of the chip are controlled with micro-commands issued by a common decoder. This decoder receives directly instructions from the outside and propagates amplified signals throughout the chip.

Due to the highly regular chip structure, full custom cells were designed to get a very compact circuit without any routing connections. The chip core can be seen as a large array of elementary cells placed side by side. This simplifies greatly the design since only a reduced version can be generated for simulation, layout verification or layout/schematic comparison.

6 Performance

Correction process including transposition errors involves a 12 machine cycle systolic step ; in other words, with a 25 Mhz clock, the computation time required for calculating the basic equation is equal to 480ns. Since an edit distance result is available every systolic cycle, more than two millions strings can be processed per second. When all processors are operating, a **1.3 Gops** (equivalent to 8 bit addition) peak performance is achieved.

The chip has been designed in a 1.5μm CMOS technology. It integrates 270,000 transistors on a 10mm $\times$ 12mm silicon area (without pads). A smaller version of the chip is presently under fabrication to validate our design methodology. It has been generated automatically by the layout generator developed in covert with the basic cells. This version includes a matrix of 3 $\times$ 12 processors allowing the efficient implementation of string comparison algorithms.

References

[1] P. A. V. Hall, G. R. Dowling, "Approximate string matching," *Comput. Surv.,* vol. 12, pp. 381-402, 1980.

[2] R. Lowrance, R. A. Wagner, "An extension of the string to string correction problem," *J. Assoc. Comput. Mach.,* vol. 22, n° 2, pp. 177-183, 1975.