Parallel Algorithms and Applications, Vol. 11, pp. 27-43
Reprints available directly from the publisher
Photocopying permitted by license only

© 1997 OPA (Overseas Publishers Association)
Amsterdam B.V. Published in The Netherlands under
license by Gordon and Breach Science Publishers
Printed in India

GLOBAL COMMUNICATIONS ON A LINEAR ARRAY ARCHITECTURE*

ALBERTO BROGGI†

Dipartimento di Ingegneria dell'Informazione, Università di Parma, I-43100 Parma, Italy

(Received 4 April 1995; In final form 8 January 1996)

This paper presents a novel interprocessor communication network which has been designed to be integrated on a specialized low-cost massively parallel system, PAPRICA-3, featuring 256 single-bit processors arranged as a linear array. This communication network is of basic importance in the implementation of communications among processors not directly connected, and can solve efficiently problems involving non-local exchange of information such as image transforms, seed-propagation, labeling of connected components, or the handling of hierarchical data structures.

A sample application, the implementation of an image transform for road markings detection, requiring global line-wise data transfers among processors is discussed and the computational complexity of its implementation on PAPRICA-3 is presented.

Keywords: Linear array; intercommunication network; hierarchical data structure; image transform

Classification Categories: C 1.2, C 2.1, I 4.5.

1 INTRODUCTION

The processing techniques based on Cellular Automata [8, 9] or Mathematical Morphology [20, 12] can be efficiently mapped on mesh-connected SIMD systems, as in the case of the low-level image processing filters. Conversely, the implementation of more complex filters (such as image transforms

* This work was partially supported by the Italian Research Council-Progetto Finalizzato Trasporti-PROMETHEUS.

† E-Mail: broggi@CE.UniPR.IT, Tel: +39-521-905707 - Fax: +39-521-905723

requiring non-local exchange of information among processors) on these systems becomes inefficient. A communication network with a topology more sophisticated than the simple *nearest neighbors* interconnections of mesh-connected systems is then required to perform non-local (global) communications efficiently. A lot of different network topologies have been investigated in the literature (shuffle, butterfly, hypercube, ... [13]), but due to their complexity, their hardware integration is expensive and thus not implementable on *low-cost* machines, such as PAPRICA-3.

In fact, the PAPRICA-3 (PArallel PRocessor for Image Checking and Analysis, version 3) [5] masively parallel SIMD architecture has been designed as a low-cost hardware engine for real-time low-level image processing. It has been developed starting from the experience gained in the implementation and use of the first PAPRICA prototype [6], composed of 256 single-bit Processing Elements (PEs) disposed on the nodes of a 2D square grid.

Some performance issues, which have been theoretically addressed in [11] and extensively tested in the design of applications [1, 4], have led to the choice of a more performing linear array architecture [19] enhanced with a simple but effective reconfigurable interprocessor communication network and a direct I/O interface to the acquisition device. The system specifications are now completely defined, and the ICs are being designed; the system is currently simulated on Unix, Linux, and DOS machines, while the complete hardware system will be available by Fall '96.

The paper is organized as follows. The next section describes the PAPRICA-3 low-cost massively parallel system and its powerful interprocessor communication network. Section 3 analyzes theoretically the problem of simultaneous non-local communications over this network and presents the implementation on PAPRICA-3 system. Section 4 discusses a sample application based on non-local data transfers, which can be efficiently implemented exploiting the interprocessor communication network. Section 5 ends the paper with some concluding remarks.

2 THE PAPRICA-3 MASSIVELY PARALLEL SYSTEM

As shown in Figure 1a, the core of the system is a dedicated SIMD cellular architecture based on a linear array of 256 identical single-bit PEs, controlled by a front-end serial processor. The array is connected to an external Image Memory via a bidirectional 256-bit data bus; therefore each memory read or write operation transfers a complete vector of 256 pixels at a time, 1 bit per pixel.

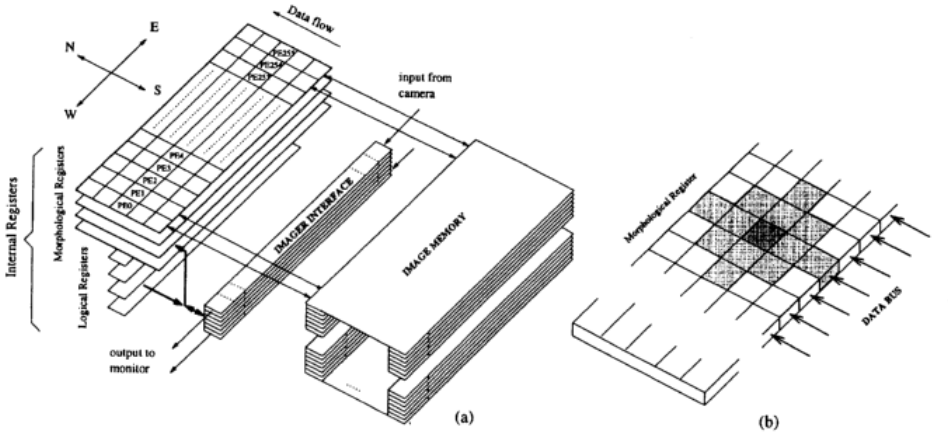

FIGURE 1 (a) structure of a 256 PEs version of PAPRICA-3 and (b) enhanced 3 × 3 neighborhood.

The rationale behind this system is that the size of the Processor Array (PA) matches exactly the width of the input image, thus reducing the PE virtualization problem, which has been proved to be a critical design issue in 2D arrays [11, 2]. The PA processes one full image line per machine cycle, whose duration ranges from 40 ns to 100 ns, depending on the specific instruction flow [10]. Data are transferred into the PEs internal registers, processed, and explicitly stored back again into the external memory according to a RISC-oriented processing paradigm. Since the instruction set is based on morphological operators, the result of an operation depends, for each PE, on the values of pixels in a given neighborhood (enhanced 3 × 3, as shown in Figure 1b). Data from EAST and WEST directions may be obtained by direct connection with neighboring PEs while all other directions correspond to data of previous (N,NE,NW,NN) or future (S,SE,SW,SS) data lines. For this reason a number of processor registers (*Morphological Registers, MOR*) have a structure which is more complex than that of a simple memory cell and are in reality composed of 5 1-bit cells with a S→N shift register connection. Besides the MOR, each PE owns also a number of *Logical Registers (LOR)* which may be used for intermediate storage and for all operations which do not require the use of neighboring values.

The system features also a serial-to-parallel I/O device, called *IMager INterface* (IMIN), connected to a conventional camera and to a monitor. While a line is processed, the IMIN behaves like a shift-register, serially loading the following image line from the camera. At the end of the processing, the PA loads the following line from the IMIN and eventually stores the results back again into the IMIN for the successive output on a monitor: in fact, during the

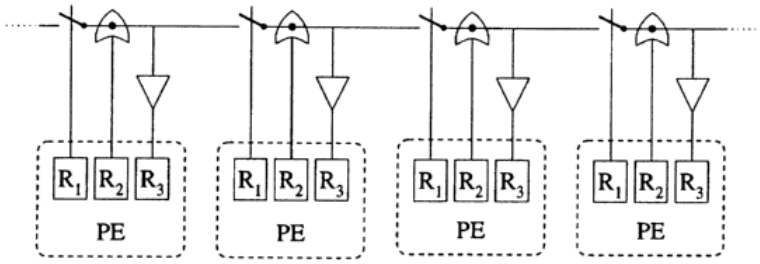

FIGURE 2 Interprocessor Communication Network (ICN): for each PE, Register R_1 drives its ICN switch; Register R_2 sends its value over the network; and Register R_3 collects the resulting value from the network.

data acquisition process, the IMIN serially loads the incoming data from the camera and simultaneously outputs the processed data serially to a monitor.

Finally, an interprocessor communication mechanism is available for exchanging information among PEs which are not directly connected: the Interprocessor Communication Network (ICN), shown in Figure 2. It allows global and multiple communications among components of different subsets of the PA (clusters of adjacent PEs) and its interconnection topology is fully and dynamically programmable. In fact, each PE drives a switch that enables or disables the communication (in wired-or) between itself and its left neighbor; the PEs can thus be dynamically grouped into clusters, and each PE can broadcast its value to the whole cluster within a single instruction. This architectural feature can be extremely useful in algorithms involving *seed-propagation* techniques, and in the emulation of pyramidal (hierarchical) processings [7, 18].

3 LINE-WISE GLOBAL COMMUNICATIONS

In a single instruction each PE can access the information stored in its neighborhood (composed of the standard Moore 3×3 neighborhood and the 4 cells at a distance 2 in the 4 main directions), as shown in Figure 3a; this allows to perform any Cellular Automata based operation [8, 9]. Figure 3b and 3c show a set of data transfers between cells not directly connected: this kind of non-local data transfers are called *global communications*. In the first case (Figure 3b) the distance between source and destination processors is equal for every PE and thus several shift operations allows to perform simultaneously all the communications, using the neighborhood PE interconnections. Conversely, Figure 3c shows a set of communications whose source-destination distance is not constant. This kind of communications can

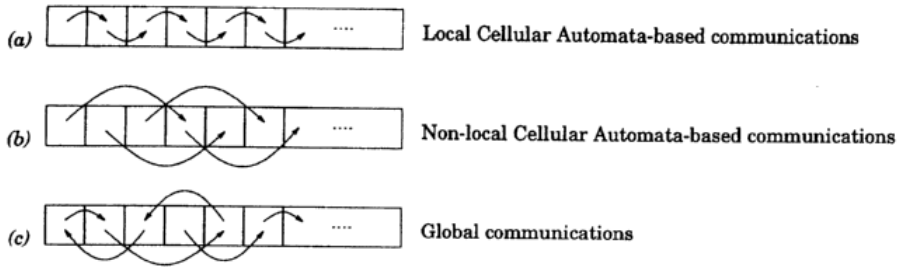

FIGURE 3 Different kinds of interprocessor communications.

be efficiently performed by the ICN network, but in general the ICN cannot perform all global communications simultaneously. As an example, Figure 4a presents a set of communications which cannot be performed concurrently because the ICN subsystem is formed by a single communication line, which can be eventually partitioned into independent segments thanks to the set of switches driven by the PEs. Note that each PE can be the source for one or more communications, but it can be the destination of a single communication only. Thus a few communication steps are required to accomplish the whole image line reorganization, each step involving only the communications that can be performed concurrently. An algorithm is thus required in order to determine the optimal partitioning of the set of communications into the smallest number of subsets. As an example, an optimal partitioning of the set shown in Figure 4a is depicted in Figure 4b.

3.1 Concurrent Communications Over the ICN

Let us consider a system composed of a set of N processing elements p_j, with $j = 0, 1, \ldots N-1$, which can send or receive data over the ICN communication line, that can be eventually partitioned into independent segments thanks to N switches, each driven by a single processor (see Figure 2). A communication $C(a, b)$ between processor p_a and p_b, with $a \leq b$ (where p_a is not necessarily the source and p_b the destination), requires the allocation of a portion of the communication line: more precisely the *cluster* of adjacent processors $\{p_a, p_{a+1}, \ldots, p_b\}$ must be allocated and uniquely reserved to that communication.

Two communications (or their corresponding clusters of processors) $C(a, b)$ and $C(d, e)$ can be ordered, the ordering rules being as follows:

$$\begin{cases} C(a,b) < C(d,e) \Leftrightarrow a < d, \\ C(a,b) = C(d,e) \Leftrightarrow a = d, \\ C(a,b) > C(d,e) \Leftrightarrow a > d. \end{cases} \tag{1}$$

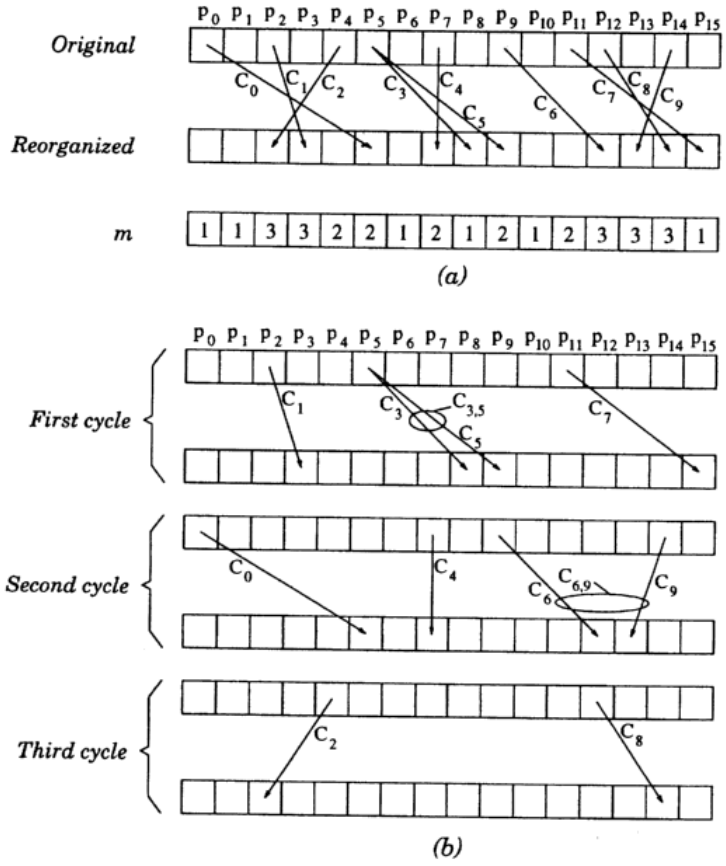

FIGURE 4 Example of line-wise communications on a 16 PE linear architecture.

Two communications $C(a, b)$ and $C(d, e)$ are said to be *adjacent* iff $d = b + 1$. Two communications $C(a, b)$ and $C(d, e)$ are said to be *compatible* if and only if there is no overlapping between their two clusters of processors $\{p_a, p_{a+1}, \ldots, p_b\}$ and $\{p_d, p_{d+1}, \ldots, p_e\}$ (i.e. they have a null intersection); namely

$$C(a, b) \text{ and } C(d, e) \text{ are } \textit{compatible} \text{ iff } \begin{cases} b < d, & \text{when } C(a, b) < C(d, e), \\ e < a, & \text{when } C(a, b) > C(d, e), \end{cases} \tag{2}$$

and cannot be compatible if $C(a, b) = C(d, e)$. Following this definition, a *set of communications* is said to be *compatible* if any couple of its elements is also compatible. Two adjacent communications $C(a, b)$ and $C(b + 1, e)$ are compatible and can thus be performed simultaneously; for sake of simplicity, in the following discussion they will be replaced by a single communication $C(a, e)$. In the example of Figure 4 $C_6 = C(9, 12)$ and $C_9 = C(13, 14)$ are replaced by $C_{6,9}$. Two communications $C(a, b)$ and $C(d, e)$ sharing the

same source processor ($C_3 = C(5, 8)$ and $C_5 = C(5, 9)$ in the example of Figure 4) can be performed simultaneously, and thus replaced by a single communication ($C_{3,5}$) which requires the allocation of the cluster of processors $\{p_{\min(a,d)}, \ldots, p_{\max(b,e)}\}$, given by the union of the two previous original clusters.

Given a set of s communications $C_i \triangleq C(a_i, b_i)$ with $i = 0, 1, \ldots, s-1$, each processor p_j is assigned a value $m_j \leq s$ which represents the number of times the allocation of the processor itself is requested by the whole set of communications C_i. The maximum value $w = \max_j m_j$ is called the *multiplicity* of the set of communications C_i. In the example of Figure 4 it is $w = 3$.

It is obvious that a set C_i with multiplicity w cannot be exhaustively partitioned into q non-overlapping (with null intersection) sets of compatible communications, when $q < w$.

THEOREM *Given a communication line with N processing elements p_j, $j = 0, 1, \ldots, N-1$, and given any set of s communications C_i, $i = 0, 1, \ldots, s-1$, with multiplicity w, it is always possible to partition exhaustively the set of communications C_i into w non-overlapping sets of compatible communications.*

Proof Let us consider the following set of processors M, defined as:

$$M = \{p_j | m_j = w, with j = 0, 1, \ldots, N-1\}. \tag{3}$$

In general, M is formed by $h > 0$ adjacent clusters of processors:

$$M = \{\underbrace{p_{x_0}, p_{x_0}+1, \ldots, p_{y_0}}_{\text{Cluster \#1}}, \underbrace{p_{x_1}, p_{x_1}+1, \ldots, p_{y_1}}_{\text{Cluster \#2}}, \ldots, \underbrace{p_{x_{h-1}}, p_{x_{h-1}}+1, \ldots, p_{y_{h-1}}}_{\text{Cluster \#}h}\}, \tag{4}$$

where $h = C(M)$ is called *connectiveness* of set M.

Each cluster of processors $T_k = \{p_{x_k}, p_{x_k+1}, \ldots, p_{y_k}\}$, with $k = 0, 1, \ldots, h-1$, detects a subset H_k of communications C_i such that:

$$H_k = \{C_i = C(a_i, b_i) | \{a_i, a_{i+1}, \ldots, b_i\} \cap T_k \neq \emptyset, \text{ for each } i = 0, 1, \ldots, s-1\}, \text{ with } k = 0, 1, \ldots, h-1. \tag{5}$$

Since two adjacent communications are handled as a single communication, the allocation of two adjacent processors p_j and p_{j+1} with $m_j = m_{j+1}$ is requested by the same set of communications. Thus the allocation of processors $\{p_{x_k}, p_{x_k+1}, \ldots, p_{y_k}\}$, with $k = 0, 1, \ldots, h-1$ of cluster T_k is requested by each one of the elements of H_k.

In the example of Figure 4, it is:

$$\begin{cases} M = \{p_2, p_3, p_{12}, p_{13}, p_{14}\}, \\ h = 2, \\ H_0 = \{C_0, C_1, C_2\}, \text{ and } H_1 = \{C_{6,9}, C_7, C_8\}. \end{cases} \tag{6}$$

The proof (by induction) is divided into 3 steps: (a) first the theorem is proven for $h = 1$, (b) then for $h = 2$, and finally (c) assuming it valid for $h = 1, 2, \ldots, n-1$ it is proven for $h = n$.

(a) $h = 1$:
in this case, C^1 is formed by just one of the elements of H_0; the multiplicity of the remaining set of communications C^2 is then reduced by 1.

(b) $h = 2$:
when $h = 2$, two different cases arise:

1. the intersection between H_0 and H_1 is not null;
   in this case C^1 is formed by just one of the elements of the intersection between H_0 and H_1; the multiplicity of the remaining set of communications C^2 is then reduced by 1.
2. the intersection between H_0 and H_1 is null;
   in this case H_0 and H_1 represent two disjoint sets of communications, with connectiveness $C(H_0) = C(H_1) = 1$, which have been previously considered in step (a) (when $h = 1$).

(c) $h = n$:
two different cases arise:

1. the intersection among the n sets H_k, with $k = 0, 1, \ldots, n-1$ is not null;
   in this case C^1 is formed by just one of the elements of the intersection among the H_k sets; the multiplicity of the remaining set of communications C^2 is then reduced by 1.
2. the intersection among the n sets H_k, with $k = 0, 1, \ldots, n-1$ is null; in this case the n sets can be partitioned into two disjoint sets H' and H'', with connectiveness $C(H'), C(H'') < n$, whose independent partitioning is assumed demonstrated by the induction process. □

COROLLARY *It is always possible to determine a partitioning of any set of s communications C_i with multiplicity w in two disjoint subsets C^1 and C^2, where the former is a set of compatible communications (a set with multiplicity 1), and the latter is a set with multiplicity $w - 1$.*

Discussion:

a) The major result is that the only parameter that influences the communication time is w, which is anyway somewhat dependent from s. However, it does not necessarily follows that an increment in the number of communications s corresponds to an increment in the set multiplicity w. In the worst case (namely when all compatible sets are formed by a single communication) it is $w = s$, but in general it is $w < s$.

b) Since the determination of the w sets of compatible communications requires the complete knowledge of the whole set of communications, it can be performed by an external processor[1] only. Thus, this mechanism is extremely efficient in the simulation of pyramidal interconnections between the PEs [7, 18], in the emulation of the non-uniform image sampling of the human visual system [21, 16], or, in general, in the solution of problems involving a fixed set of communications known at compile time, since the determination of the sets of compatible communications is performed off-line. Conversely, its effectiveness decreases for the communication of values whose source and/or destination processors are known only at run-time, since the front-end processor must determine the sets of compatible communications at run-time.

3.2 Determining the w Sets of Compatible Communications

The algorithm used by the front-end to determine the w sets of compatible communications is directly derived from the corollary discussed in section 3.1, which states that with a single scanning it is possible to split the set of s communications (with multiplicity w) into a subset of compatible communications and a set with multiplicity $w - 1$; the problem is thus solved after w iterations of the above procedure. Figure 5 shows the algorithm used to determine both the multiplicity w of the initial set and the w sets of compatible communications.

Due to the specific implementation of the labeling procedure, the determination of the compatibility between a single communication and a set has complexity 0(1); consequently the total algorithm complexity is $0(s.w)$.

3.3 The SIMD Communication Algorithm

Now let us focus on the SIMD implementation of the communication algorithm. As explained above, the execution of a single data transfer between

[1] For example a front-end host processor.

```
count = 0;                          Reset the counter of already labeled communications.
w = 0;                              Set the initially unknown multiplicity w to 0.
for (i=0; i<s; i++) label[i] = 0;   Mark all communications as unlabeled.
repeat {                            Repeat for each subset.
  w++;
  for (i=0; i<s; i++) {             Scan all communications.
    if ( (!label[i]) && (c[i] is compatible with set[w]) ) { If the i-th communication has not
                                    been labeled yet and if it is compatible with the current
                                    w-th set,
      label[i] = w;                 then label it
      count++;                      and increment the counter of already labeled
                                    communications.
    } /* if */
  } /* for */
until (count==s);                   Loop until every communication has been labeled.
```

FIGURE 5 The algorithm used by the front-end to determine the w sets of compatible communications.

a source and as many destinations as required needs the exclusive allocation of a cluster of adjacent PEs. Thus, for each set of compatible communications, three different binary constants are defined:

- **Source:** specifying the PEs which act as sources in the current set of communications;
- **Path:** defining the grouping of PEs; since each PE drives a switch that connects itself to its left neighbor, the leftmost PE in a cluster must hold a zero value;
- **Destination**: identifying the destination PEs which must read and store in their local memory the value collected from the ICN network.

The algorithm used to perform the whole set of s communications (w cycles) is shown in Figure 6.

Figure 7a and 7b present an oversampling and an undersampling mapping respectively; in both cases the data movements can be exhaustively

```
result = 0;                         Reset the final result since each partial result (one for
                                    each set of compatible communications) will be
                                    accumulated in OR.
for (i=0; i<w; i++) {               Loop over the w cycles.
  for (j=0; j<n; j++) {             Loop over the n bits per pixel of the input image.
    s = source[i] & image[j];       Reset all non-source PEs, thus keeping only one value
                                    for each cluster.
    commout s, path[i];             Send the value of s over the ICN which is configured
                                    according to path.
    commin d;                       Store in d the final status read from the ICN network.
    result[j] | =d & destination[i]; Accumulate in OR only the values collected from the
                                    ICN which correspond to destination PEs.
  }
}
```

FIGURE 6 The SIMD algorithm used to perform multiple global communications over the ICN.

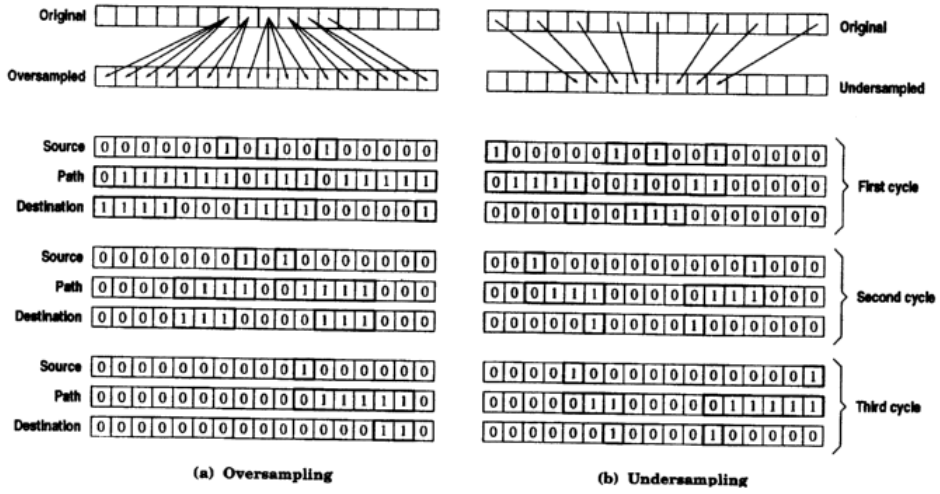

FIGURE 7 Line-wise global communications.

partitioned into 3 subsets of compatible communications (requiring 3 cycles of communications).

4 A SAMPLE APPLICATION: REAL-TIME ROAD MARKINGS DETECTION

Let us consider the problem of road markings detection in images acquired by a camera installed on a moving vehicle. Figure 8a presents an image acquired from the MOB-LAB land vehicle (the MOBile LABoratory integrating the results of the Italian activities within the Eureka PROMETHEUS Project). Due to the perspective effect, in the acquired image the road markings' width changes according to their distance from the camera. More precisely, the perspective effect makes image pixels having different meanings (and thus

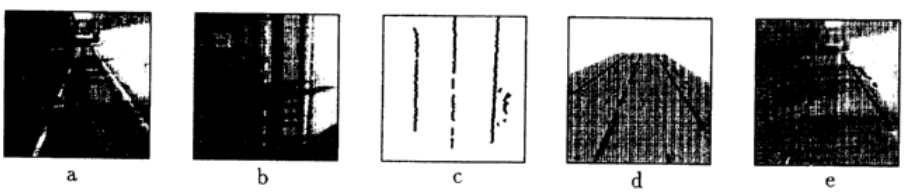

FIGURE 8 The detection of road markings through the removal of the perspective effect: (a) input image; (b) reorganized image, obtained by non-uniform resampling of (a); (c) result of the line-wise detection of black-white-black transitions in the horizontal direction; (d) reintroduction of the perspective effect (the gray areas represent the portion of the image shown in (c)); (e) superimposition of (d) onto a brighter version of the original image (a).

carrying different properties) depending on their position within the image. For example, a pixel in the lower part of the image shown in Figure 8a represents a few cm^2 of the road, while a pixel in the middle of the same image represents a few tens of cm^2, or even more. The determination of the road markings' width with pattern matching techniques is then based on matchings with templates whose size varies according to the image coordinates; thus the implementation of this process on SIMD machines (where all pixels are processed according to the same rule) cannot achieve a high efficiency.

Conversely, the removal of the perspective effect, if possible, would cause each pixel to represent the same portion of the road (see Figure 8a). In this way the road markings' width becomes invariant with their position within the image, and thus also the feature extraction procedure (i.e. the size and shape of the matching template) becomes independent of the pixel position; consequently it can be implemented extremely efficiently on SIMD systems.

Beside being efficiently implementable on any mesh-connected SIMD massively parallel system, the processing of this new image, where the perspective effect has been removed, can solve effectively one of the main problems of vision-based road detection systems [14, 15]: the presence of shadows on the road region. In fact road markings are now represented by quasi-vertical lines of constant width that can be easily extracted through a simple morphological algorithm. The result of this morphological processing [3] is shown in Figure 8c, while Figure 8e presents its superimposition onto the original image.

4.1 Removing the Perspective Effect

The procedure aimed to remove the perspective effect from the acquired image is a transform, that reads the incoming image and resamples it, remapping each pixel toward a different position and producing a new 2-dimensional array of pixels in a new domain. The new image represent a top view of the road region in front of the vehicle.

Two Euclidean spaces are thus defined:

- $W = \{(x, y, z)\} \in E^3$ representing the $3D$ world space (*world-coordinate system*), where the real world is defined;
- $I = \{(u, v)\} \in E^2$ representing the $2D$ image space (*screen-coordinate system*), where the 3D scene is projected.

The image acquired by the camera belongs to the I space, while the reorganized image is defined as the $z = 0$ plane of the W space (according to the

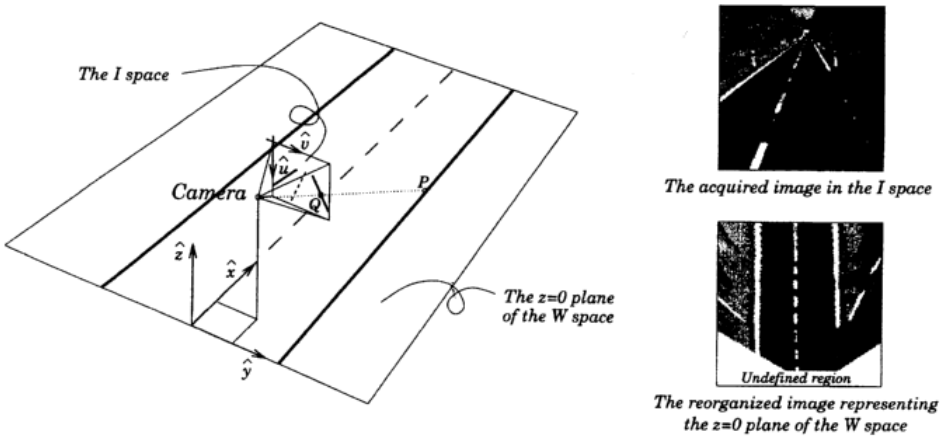

FIGURE 9 The relationship between the two coordinate systems: an original image and the result of its reorganization; due to the specific camera position and to its finite angolar aperture, the lower portion of the reorganized image is undefined.

assumption of a *flat* road). Figure 9 shows the relationships between the two spaces W and I.

The reorganization process associates to each pixel $P = (x, y, 0)$ of the reorganized image its corresponding pixel $Q = (u, v)$ in the original one: namely, as shown in Figure 9, it projects the acquired image onto the $z = 0$ plane of the 3D world space W, acting as the dual of a *ray-tracing* algorithm [17].

In order to generate a 2D view of a 3D scene, a *viewpoint, viewing direction, aperture*, and *resolution* must be specified [17]. Let us make the following assumptions:

- *viewpoint*: the camera is placed in $C = (l, d, h) \in W$;
- *viewing direction*: the optical axis $\hat{o}$ is determined by the following angles:

  $-\bar{\gamma}$: the angle formed by the projection (defined by versor $\hat{\eta}$) of the optical axis $\hat{o}$ on the plane $z = 0$ and the x axis (as shown in Figure 10a);

  $-\bar{\theta}$: the angle formed by the optical axis $\hat{o}$ and versor $\hat{\eta}$ (as shown in Figure 10b);

- *aperture*: the camera angolar aperture is 2α;
- *resolution*: the camera resolution is $n \times n$.

In [3] the mathematical details leading to an approximated mapping $I \rightarrow W$ are presented. Given the coordinates (u, v) of a generic point Q in the I space, equations (7) return the coordinates $(x,y,0)$ of the corresponding point P in the

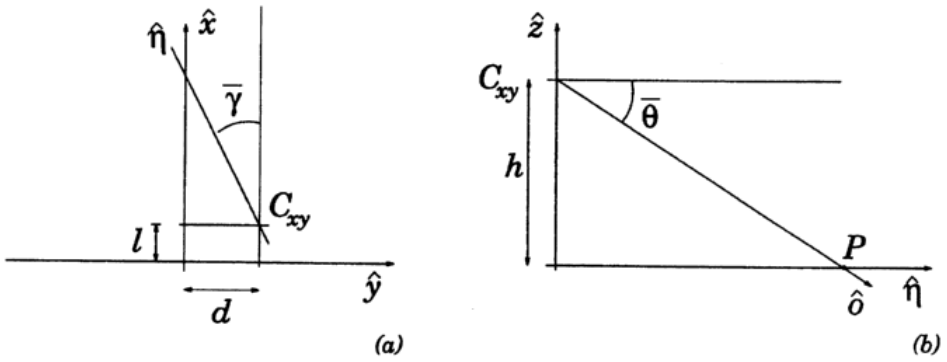

FIGURE 10 (*a*) the xy plane in the W space and (*b*) the $z\eta$ plane, assuming the Origin translated onto the projection C_{xy} of C on $z = 0$.

W space (see Figure 9).

$$\begin{cases} x(u,v) = \dfrac{h}{tg\left[(\bar{\theta}-\alpha)+u\dfrac{2\alpha}{n-1}\right]}\cos\left[(\bar{\gamma}-\alpha)+v\dfrac{2\alpha}{n-1}\right]+l \\ y(u) = \dfrac{h}{tg\left[(\bar{\theta}-\alpha)+u\dfrac{2\alpha}{n-1}\right]}\dfrac{\sum_{v=0}^{n-1}\sin\left[(\bar{\gamma}-\alpha)+v\dfrac{2\alpha}{n-1}\right]}{n}+d \\ z = 0 \end{cases} \tag{7}$$

From equations (7) it descends that the set of pixels of the acquired image with u = constant (thus belonging to the same image line) are mapped toward positions in the W space with the same y coordinate. This means that the reorganization can be performed linewise, each line independently of the others.

4.2 Parallel Line-wise Image Reorganization on PAPRICA-3

The reorganization of an image requires the determination of the sets of compatible communications for each one of the n image lines. Since the camera calibration parameters are known at compile-time (and are considered fixed), this is done once for all, off-line, at the beginning of the processing; the results (the 3 Source, Path, and Destination constants for each set of compatible communications) are stored into PAPRICA-3 memory to be use as a look-up table.

Thanks to the PAPRICA-3 system-level simulator (written in standard C language and currently running on Unix, Linux, and DOS machines) it is

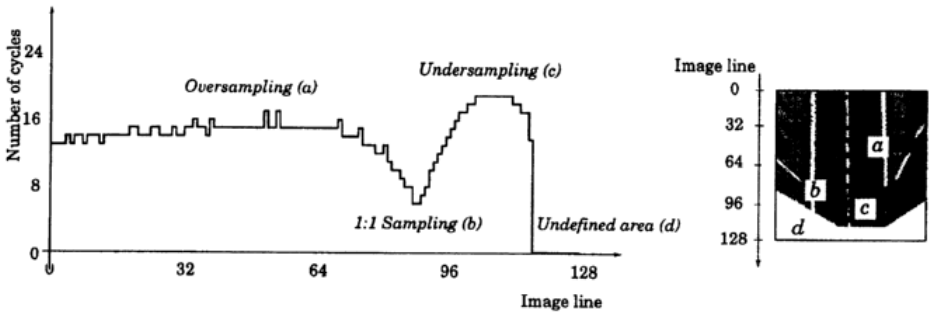

FIGURE 11 Number of communication cycles required to reorganize a 128 pixel line as a function of the position of the image line. In phase (a) (corresponding to a heavy oversampling) the number of iterations remains almost constant; it is reduced to a minimum when a *quasi* 1:1 mapping is performed (b); and it grows up again in the undersampling phase (c); finally it is reduced to zero in the lowest *undefined* region (d).

possible to derive the time required for the execution of a single cycle of simultaneous communications. The algorithm shown in Figure 6 has been implemented in PAPRICA-3 Assembly language; assuming that the Source, Path, and Destination binary constants have been previously loaded into the image memory (as mentioned above), a single cycle of compatible communications requires the execution of about 30 Assembly instructions. Since in the worst case [10] a single instruction takes 100 ns, a set of compatible communications is performed in about 3 μs. In the reorganization of a 128 $\times$ 128 pixel image, as shown in Figure 11, a maximum of 19 cycles per line is required; thus each single image line is reorganized in less than 60 μs. On the other hand, the average time required by a Sparc-based front-end processor to determine the sets of compatible communications is about 5 ms.

A commercially available off-the-shelf camera (512$\times$512 pixels, 8 bit/pixel, 25 Hz) acquires a single line (512 pixels) in 80 μs; thus, thanks to the capability to overlap data I/O and processing, PAPRICA-3 allows to perform the complete line reorganization in real-time during the acquisition of the subsequent image line.

5 CONCLUSIONS

In this work the massively parallel architecture PAPRICA-3, based on a linear array of 256 PEs, has been presented. PAPRICA-3 features a low-cost interprocessor communication mechanism (ICN) that allows to exchange information among the processors which are not directly connected. This architectural enhancement, whose main characteristics have been deeply discussed, allows

to solve efficiently a number of common problems involving global communications among PEs.

A sample application, the implementation of an image transform for road markings detection, has been presented: the ICN allows to remove the perspective effect in real-time during the acquisition of the image, causing no additional computational overhead. The computational complexity of both the offline algorithm for the determination of the sets of compatible communications and the communication algorithm itself have been derived.

References

[1] Adorni, G., Broggi, A., Conte, G. and D'Andrea, V. (1996). "Real-Time Image Processing for Automotive Applications". In P. A. Laplante and A. D. Stoyenko, editors, *Real-Time Image Processing: Theory, Techniques and Applications*. IEEE Press, 161-194.

[2] Broggi, A. (May 2-6 1994). "Performance Optimization on Low-Cost Cellular Array Processors". In *Proceedings MPCS - IEEE International Conference on Massively Parallel Computing Systems*, 334-338, Ischia, Italy. IEEE Computer Society - EuroMicro.

[3] Broggi, A. (September 25-26 1995). A Massively Parallel Approach to Real-Time Vision-Based Road Markings Detection. In I. Masaky, editor, *Proceedings IEEE Intelligent Vehicles'95*, Detroit. IEEE Computer Society, 84-89.

[4] Broggi, A. (February 1995). "Parallel and Local Feature Extraction: a Real-Time Approach to Road Boundary Detection". *IEEE Transactions on Image Processing*, **4**(2), 217-223.

[5] Broggi, A., Conte, G., Burzio, G., Lavagno, L., Gregoretti, F., Sansoè, C. and Reyneri L. M. (November 13-16 1994). "PAPRICA-3: A Real-Time Morphological Image Processor". In *Proceedings ICIP - First IEEE International Conference on Image Processing*, 654-658, Austin, TX. IEEE Computer Society.

[6] Broggi, A., Conte, G., Gregoretti, F., Sansoè, C. and Reyneri, L. M. (May 2-6 1994). "The PAPRICA Massively Parallel Processor". In *Proceedings MPCS - IEEE International Conference on Massively Parallel Computing Systems*, 16-30, Ischia, Italy. IEEE Computer Society — EuroMicro.

[7] Cantoni, V. and Ferretti M. (1993). *Pyramidal Architectures for Computer Vision*. Plenum Press, London.

[8] Culik II, K., Hurd, L. P. and Yu, S. (1990). "Computation theorethic aspects of Cellular Automata". *Physica D*, 45, 431-440.

[9] Farmer, D., Toffoli, T. and Wolfram, S. editors. (March 1983). *Cellular Automata*, Amsterdam, North-Holland.

[10] Gregoretti, F., Intini, F., Lavagno, L., Passerone, R. and Reyneri, L. M. (January, 24-26 1996). "Design and Implementation of the Control Structure of the PAPRICA-3 Processor". In *Proceedings 4rd EuroMicro Workshop on Parallel and Distributed Processing*, Braga, Portugal. IEEE Computer Society — EuroMicro. In press.

[11] Gregoretti, F., Reyneri, L. M., Sansoè, C., Broggi, A. and Conte, G. (October 25-27 1993). "The PAPRICA SIMD array: critical reviews and perspectives". In L. Dadda and B. Wah, editors, *Proceedings ASAP'93 - IEEE Computer Society International Conference on Application Specific Array Processors*, 309-320, Venezia, Italy. IEEE Computer Society — EuroMicro.

[12] Haralick, R. M., Sternberg, S. R. and Zhuang, X. (1987). "Image Analysis Using Mathematical Morphology". *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **9**(4), 532-550.

[13] Hwang, K. (1993). *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. McGraw-Hill, Inc., New York.

[14] Jochem, T. M. and Baluja, S. (1993). "Massively Parallel, Adaptive, Color Image Processing for Autonomous Road Following". In H. Kitano, editor, *Massively Parallel Artificial Intelligence*. A III Publishers in cooperation with MIT Press.

[15] Kenue, S. K. (1994). "Correction of Shadow Artifacts for Vision-based Vehicle Guidance". In *Proceedings of SPIE - Mobile Robots VIII*, **2058**, 12-26.

[16] Mead, C. A. (1982). "Neuromorphic Electronic Systems". *IEEE Proceedings*, **78**(10), 1629-1636.

[17] Newman, W. M. and Sproull, R. F. (1981). *Principles of Interactive Computer Graphics*. McGraw-Hill, Tokyo.

[18] Rosenfeld, A. (1984). *Multiresolution Image Processing and Analysis*. Springer Verlag, Berlin.

[19] Schmitt, L. A. and Wilson, S. S. (May 1988). "The AIS-5000 Parallel Processor". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10**(3), 320-330.

[20] Serra, J. (1982). *Image Analysis and Mathematical Morphology*. Academic Press, London.

[21] Zavidovique, B. and Fiorini, P. (1994). "A Control View to Vision Architectures". In V. Cantoni, editor, *Human and Machine Vision: Analogies and Divergencies*, 13-56. Plenum Press.