Akademie der Wissenschaften der DDR
Zentralinstitut für Kybernetik und Informationsprozesse (ZKI)

Basislaboratorium für Bildverarbeitung und Computergraphik

EFFECTIVE IMAGE PROCESSING USING THE SPECIAL PURPOSE PROCESSOR GIPP

Fritzsch, K., Kutschke, G.,
Roesler, U., Schwarze, G.

G. Heinz
Königsheideweg 234
O - 1197 Berlin

Preprint No. 10 (January 1988)
enlarged and revised I/1990

Preprint

Basic Laboratory of Image Processing
and Computer Graphics

Базовая лаборатория по обработке
изображений и машинной графике

Contents:

1. Introduction

Neighborhood processing is one of the most widely used kinds of parallelism in image processing. The degree of parallelism depends on the size of the neighborhood. Very popular are 3x3-neighborhood processors. Some theoretical and practical reasons for it may be reminded:

- almost all basic algorithms of preprocessing can be implemented within a 3x3-window,
- using methods of image algebra preprocessing within larger windows can be reduced to 3x3-window processing under weak constraints,
- special purpose hardware for typical image size and gray level resolution may be developed using a single board for one processor working at video rate.

In neighborhood processors the single pixels in an image are processed serially, therefore the time is growing linearily with image size. Nevertheless, the processing may be accelerated by building up pipelines of differently programmed processors each of them realizing one processing pass.
Massive parallelism which means parallel processing of the pixels of the image is the most effective but also most expensive kind. The upper bound for the acceleration is equal to the number of pixels of one image, i.e. to the image size.
Architectures may be evaluated on the base of the facilities for parallelization as described above. But also the organization of control and data flows are an important feature. A strict separation of both flows is presumably the best solution for an advanced image processor.
A decisive stage is that step in the processing where the transition from the low-level raster representation to intermediate-level representation occurs. Therefore the output behavior of a low-level processor is of great importance. Advanced architectures offer possibilities to produce intermediate-level image representations as component tables, run-length codes or other contour descriptions, etc.

2. Architectural and algorithmic preliminaries

2.1 Useful architectures

Among the numerous proposals for image processors some attractive solutions have found broad acceptance and are implemented in commercially available systems.
Processors with an internal pipeline consisting of look-up tables for gray level transforms and of arithmetical-logical units for combining data from different image memories are used in some systems. For example in the image processing system A6472 the processing is logically parallel, but physically serial. It offers a high degree of flexibility for full-frame processing at the expense of demanding storage capacity for several images (Wilhelmi, 1983; Kempe, 1982).
Neighborhood processors are built for some classes of algorithms, especially for convolution and morphology. A(n external) pipeline of differently programmed processors is used in the most prominent device of this kind, the Cytocomputer (Sternberg, 1981). The GIPP processor described later on is the basic structure for a very similar architecture.

The single convolution steps may be accelerated by signal processors which are supporting linear transforms of the kind

$$d = a * b + c.$$

Array processors are mainly used for massively parallel processing (Duff, 1983 ; Parallel Computing, 1987). They may be ascribed to high-performance systems whereas the other architectures decribed before are typical for medium-performance systems. In the future, the development may be fertilized by the systolic approach because of its good utilization of processing resources and of the possibilities of effective VLSI design .
Programmability will remain one of the most important features for image processors.
Further, an effective access to the image memory is neccessary. But fully exploiting the regular two-dimensional nature of an image is leading to demands for sophisticated address controllers. These difficulties may be alleviated when the processor is constructed in such a way that it accepts serial input resulting from raster-scanning the images. This line-by-line raster scan is also the mode how images are received from sensors. Obviously, in the case of neighborhood processing several neighboring lines must be hold in a buffer memory or in shift registers.

2.2 Some basic methods of neighborhood preprocessing

Preprocessing or low-level processing is understood as processing iconic image representations comprising mainly two basic methods: single-pixel processing and neighborhood processing.
Into the concept of single-pixel processing, such methods are included as gray level transforms of all kinds, among them thresholding, further pseudocolorization, combining images and others.
In neighborhood processing the new value of one pixel in an iconic representation depends not only on the old value of the pixel at this location but also on the values of neighboring pixels.
Basic processes are:

a) Cleaning and restoring
   Using a model of image generation and of influence of noisy perturbations erroneous pixel values are corrected assuming that useful information is retained in the neighborhood.
b) Edge finding
   Pronounced changes in the gray level distribution within the image can be found comparing gray levels in a local surrounding of one pixel.
c) Thinning and contrasting
   Smooth edges or blurred lines can be sharpened by locating the pixels with steepest descents or with extrema of gray level.

The mathematical base is given by convolution within a two-dimensional window, by sorting or comparing pixel values in dependence of certain criteria, by mask matching for deriving conditions for application of local operators, and so on.
The mostly used size of windows is 3x3, but sometimes windows up to a size of 11x11 are applied.

2.3 From low-level to intermediate-level processing

Intermediate level in image processing means that the image is no longer represented in a sensor-based iconic image but in a more object-oriented manner, as in contour- or region-oriented decription. Such decriptions are possible in numerical form on the base

of rasters, but more suited is the symbolic form of lists or chains. They are interesting for intermediate-level processing. Nevertheless, the transform to intermediate level is seldom supported by image processors if we exclude statistical operators as histograms e. g., which do not preserve any information of geometrical nature.
The question is whether there are GIPP-implementable functions which fulfill the demands of effectively transforming low-level to intermediate-level images .
One important remark is to be made concerning the representation of edges. In 3x3-window convolution by a Sobel operator e.g. the centre-pixel receives the output of a local gradient. In the GIPP processor the so-called two-point gradient is calculated. This is the maximum value of the differences of the gray values from the centre-pixel to the neighboring ones and therefore an edge value. The approximation consists in using the centre-pixel location for storing the maximum edge value. This has consequences concerning thinning and thickening as described later on.

3. Principles of GIPP-processing

3.1 Basic computing algorithm

The principles of GIPP-processing can be described in the following three steps:

1. Given is a set of a finite number k of image processing algorithms.
2. It is possible, to map this restricted class of image processing algorithms onto a single basic computing algorithm with k parameters microprograms or for this restricted class of image processing algorithms. This basic computing algorithm must be well adapted to an efficient implementation.
3. This basic computing algorithm can be efficiently implemented in the hardware structure GIPP, which is programmable for a number k of image processing algorithms.

In this sense a GIPP can be considered as a basic structure for image processing leading to the concept of GIPP-implementable image processing functions. At present all work is related to a structure using the raster-scan approach and a 3x3-window with the characteristic movement of the centre-pixel.
In general, the computation of the new centre-pixel value consists in selecting one of the gray values inside of the window. The selection rule is determined by the given processing function. Furthermore, masks can be set. If the mask is matched, the computation of the new centre-pixel value is carried out, if not, the centre-pixel remains at its former value.
Usually the GIPP works in a bit-plane manner. One iteration step is needed for each of the M bit planes. This means, that every computation of a new centre-pixel value is executed by M iteration steps:

$$M = \log_2 NG,$$

M = number of necessary iteration steps,
($\triangleq$ number of bit planes),
NG = number of gray levels.

In each iteration step k (k=1...M) the available gray value range G^k is divided into a lower gray value range G^k and an upper

one G^k of equal size:

G^k	$\{ g^k_1, g^k_{1+1}, \ldots g^k_u \}$	gray value range,
G^k_1	$\{ g^k_1, g^k_{1+1}, \ldots g^k_{m-1} \}$	lower gray value range,
G^k_u	$\{ g^k_m, g^k_{m+1}, \ldots g^k_{u-1} \}$	upper gray value range,
$g^k_m = (g^k_1 + g^k_u + 1)/2$		medium gray value of the gray value range of the k-th iteration step

g^k_1 and g^k_u are the lowest and highest gray values resp. in the k-th iteration step.
By applying the selection rule it is checked whether the lower or the upper gray value range contains the expected value. The gray value range found in the (k-1)-th step is the available gray value range G_k in the next iteration step. During every iteration step the gray values of all pixels inside the window are parallely compared with the actual medium gray value g^k_m.
During the i-th comparison the value of a binary variable w^k_i is generated. If the pixel value is greater than equal to the medium gray level g^k_m, them w^k_i is set to zero. If the pixel value is lower than the medium gray value g^k_m, them w^k_i is set to one:

$$w^k_i = \begin{cases} 0, \text{ if } G(i) = g^k_m \\ 1, \text{ else} \end{cases}$$

The result of the comparisons of all the N gray values inside the window with the medium gray value is a set of binary variables w^k_i of a function F. This function F realizes the selected image processsing algorithm.

$$F = f(w^k_0, w^k_1, \ldots w^k_{N-1}),$$

N = number of pixels inside the window.

It depends on the value of the function F in the k-th iteration step whether the computation in the following (k+1)-th iteration step is executed in the lower or upper gray value range:

$F = 1$	$g^{k+1}_1 = g^k_m$	$g^{k+1}_u = g^k_u$	upper gray value range
$F = 0$	$g^{k+1}_1 = g^k_1$	$g^{k+1}_u = g^k_{m-1}$	lower gray value range

Finally, after k = M iteration steps the new gray value of the centre-pixel is obtained.

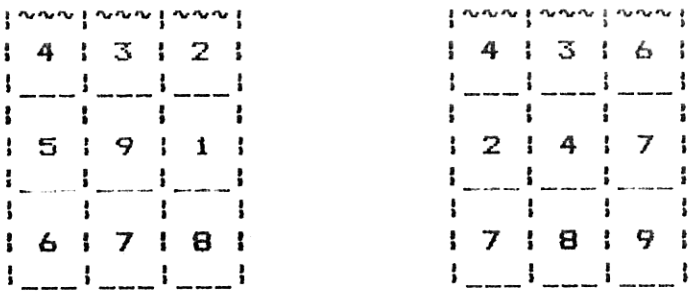
In general, the described basic computing algorithm has the following properties:
- The number M of clock periods for the evaluation of the new gray value of the centre-pixel of a given window is independent on the type of image processing function and the gray value distribution inside this window.
- The number M of clock periods is only dependent on the number NG of gray levels of the designed image processing system (M = log NG), i.e. it is equal to the number of bit planes.
- The basic computing algorithm generates at every clock period one bit of the new centre-pixel value. This means, that the different bits of a new centre-pixel value are generated in a serial manner.

- At every clock period the nine gray values of the window are compared with the value g^k. In dependence on the results of the comparisons the corresponding bit of the new centre-pixel value will be computed, that means in a parallel manner.

As an example for the basic computing algorithm, we consider the median filter for a window with N = 9 pixels and NG = 16 gray levels. The function F is in the case of median filtering:

$$F(w_0,\ldots w_8) = \begin{cases} 1, & \text{if } n>4 \\ 0, & \text{otherwise,} \end{cases}$$

where n is the number of equations for which g(i) = g holds. The pixel enumeration inside the window, an example of gray value distribution inside the window, and the results during the iteration are shown in fig. 3.1.



Median value = 6

Step	middle gray value	result of comparison	n	F
1	1000	111 111 100	2	0
2	0100	000 101 000	7	1
3	0110	100 111 000	5	1
4	0111	101 111 000	4	0
Result	0110			

Fig. 3.1 GIPP computing algorithm for median filtering

In the step k=1 the bit 1 is set , the condition n>4 is not fullfilled (F=0) and therefore the bit 1 is deleted. In the step k=2 the bit 2 is set, the condition n>4 is fullfilled (F=1) and therefore the bit 2 is not deleted. The processing of the bits 3 and 4 is the same one . This procedure corresponds to the search in a tree (fig. 3.2).

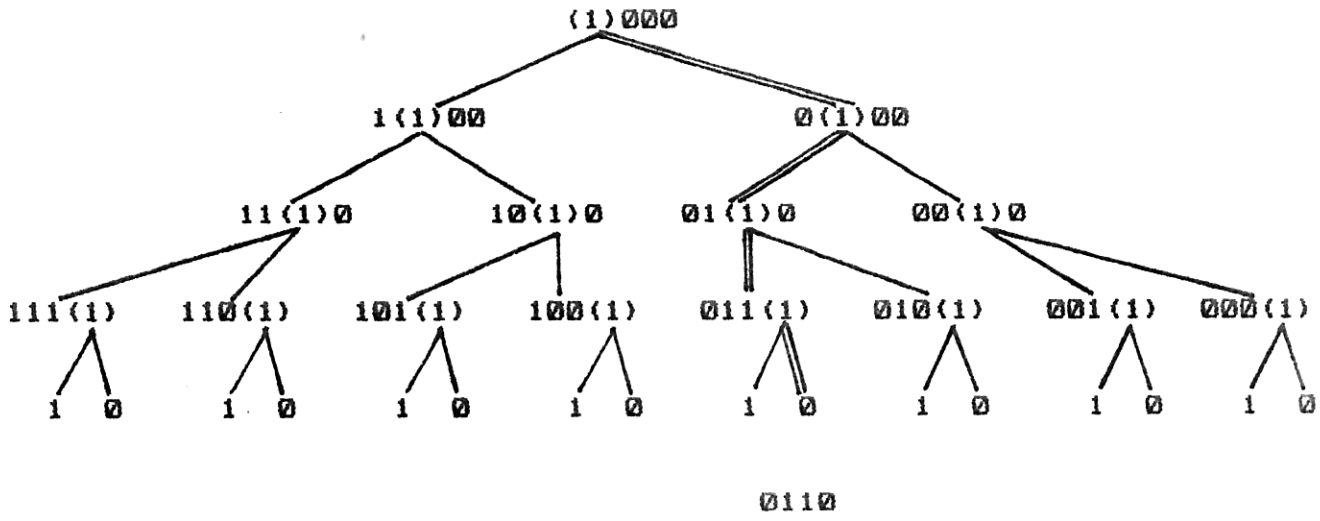


Fig. 3.2 GIPP computing algorithm mapped on a tree search (the path which is ensued from fig. 3.1 is marked)

A similar bitplane-processing algorithm for rank filtering was proposed by Ataman et. al. (1980). The disadvantage of this algorithm is the fact , that after the processing step in the bitplane i all following lower bitplanes must be tested, whatever bits in these planes must be changed. Therefore all bitplanes are coupled together.

3.2 Extensions of the basic computingalgorithm

An extension of the described basic computing algorithm is shown to be possible in the following sense:

- Setting a threshold THR
  It can be performed a comparison of all gray values g(i) inside the window with adjustable thresholds THR(i):

$$g(i) = \begin{cases} 1, \text{ if } g(i) = THR(i) \\ 0, \text{ else.} \end{cases}$$

- Setting a lookup-table
  The lookup-table can give a code symbol with the result of the thresholding.

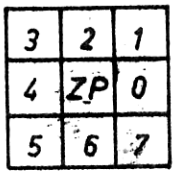
- Setting the column index.

- Setting a symbol for the end of a row.

4. GIPP-implementable functions

According to the principles of processing by the processor GIPP the computation of a new pixel value is restricted to a selection of one gray value taken from the given set of gray values in the processing window. Therefore, it is not possible to realize arithmetical operations (addition, multiplication etc.) and to compute a weighted sum, which is necessary for convolution. But

later on we shall see that functions with similar or equivalent behavior can be implemented.
The GIPP-implementable functions belong to the class of local neighborhood functions. They are defined by a special pixel, the



Central point ZP = X(0)
Neighborhood U(ZP) = {X(I); I=0,1,...,7}
Function F: {G(U), G(ZP)} → G(ZP)

Fig. 4.1: Definition of a local function for nonrecursive image processing

centre-pixel ZP, by the pixels of its (usually) square neighborhood U(ZP) and by a uniquely computable function F on the values G(X(I)) and the position of the mentioned pixels X(I). An image processing step consists in determining a new value of the central point in dependence of F. After that the window is shifted to the next pixel. The computation is done in a scan line manner with the original values of all the pixels what is called a parallel or nonrecursive computation.
The function type is given by the manner how the value of the function is selected. If the value of ZP is only composed of all the pixel values within the window regardless of their positions we speak of a window operator, if it depends of the gray values and the position of chosen pixels then it is a mask operator. Local operations process the image homogeneously, isotropically (if using a symmetrical neighborhood), within one pass parallely, and from one pass to the next sequentially. In the following we consider only 3*3 square windows for explanation. The GIPP operations are not limited to such a size and shape of the window (see chapter 5).

4.1 Rank functions

These functions belong to the class of window functions. All pixels within the window are ranked according to their values in increasing order. The rank is given by the figure R {1,....9} of the chosen rank position and the output of the operation is given by

$$G(ZP) := RANK^{9}_{R}(\{G(I)\},\ I = 0,1,....8).$$

Important cases are R = 5, 1, and 9. With R = 5 the median rank position is selected and we have the well known median function (Hodgson et al. 1985)

$$MEDIAN^{9}(\{G(I)\}) = RANK^{9}_{5}(\{G(I)\}).$$

It cleans the image, shifts its mean gray value (if not symmetrically distributed), suppresses noise, corrects pixels with extreme values (spike noise), removes thin lines of one pixel width, rounds off sharp object corners, and preserves the shape of edges. These properties are different from those of the linear

arithmetical mean (linear low pass filter) which also suppresses noise, removes small lines and corner pixels but on the expense of blurring the image.
A modification of MEDIAN is the separable median (Narenda, 1978)

$$SEPMED(\{G(I)\}) := MEDIAN^{3}(G(4+K), G(ZP), G(K)) \quad K = 0,2$$

composed of two one-dimensional medians above three values G in each case. Firstly, the line-like horizontal window is applied along the rows, and secondly, the vertical window along the columns of the image (or vica versa). The result is almost identical to the full two-dimensional median with the advantage of a simpler implementation and a faster computation if realized by a sequential algorithm. This mode of implementation cannot be extended to arbitrary rank functions.
Another modification of MEDIAN is the directional median (Ting et al. 1980)

$$DIRMED(\{G(I)\}) := \max_{K=0}^{K=3}(MEDIAN^{3}(G(4+I, G(ZP), G(K)))$$

defined by the maximum of four different 1-dimensional medians according to the four possible digital directions. It has nearly the same excellent properties as the original median but besides of it , the directional median is able to preserve digital curves (of one pixel width) and sharp corners of objects.
The repeated application of a median function with a given window size produces approximately the same effect as a median of a larger window one times applied to the same image. The behaviour of convergence is not clarified up to now.
In general all mentioned medians transform the image to a smoothed and partly restored one preserving steep gray level edges.
The other two important cases of rank operations correspond to the both extreme rank positions R = 1 and 9, that is to say, the local

$$MIN(\{G(I)\}) := RANK^{9}_{1}(\{G(I)\})$$
$$\text{and} \quad MAX(\{G(I)\}) := RANK^{9}_{9}(\{G(I)\}).$$

With the application of MIN every image object is peeled off by a shell of one pixel width whereas with the application of MAX such a shell is put on to the object. These procedures are of opposite character and called **erosion and dilatation**, respectively (with binary images we speak of shrinking and expanding). Their specific properties can be characterized by

$$MIN(MAX(MIN(P))) = MIN(P),$$
$$MAX(MIN(MAX(P))) = MAX(P) \text{ and}$$
$$MIN(MAX(P)) > P > MAX(MIN(P)),$$

if P stands for the entire image. The equality sign is only correct, if the image contains simple and large objects with smooth borders. These functions used in sequence of dilatation - erosion result in a border smoothing by filling up bays (called closing), with the inverse sequence bays are deepened (called opening). With these procedures separate objects and their parts can be connected and thin objects taken to pieces because topology is not preserved here.
The modified extreme-value function is defined by

$$\text{MODEX}(\{G(I)\}) := \begin{cases} \text{MAX}', & \text{if } G(ZP) > \text{MAX}' \\ \text{MIN}', & \text{if } G(ZP) < \text{MIN}' \end{cases}$$

and composed of the two values MAX' and MIN' which are computed from all gray values of the processing window except that of the central point ZP. It smoothes the image and reduces noise because gray level values larger than MAX' and smaller than MIN' are normalized by the values of the true neighborhood.
A further GIPP-implementable window function is called the combined extrem-value or **enhancement** function (Kramer et al. 1975) and defined by

$$\text{ENHANCE}(\{G(I)\}) := \begin{cases} \text{MAX}', & \text{if } |G(ZP) - \text{MAX}'| < |\text{MIN}' - G(ZP)| \\ \text{MIN}', & \text{otherwise,} \end{cases}$$

where the both, above explained, extreme values are combined. The local value is simply given by the lightest or darkest gray value in its neighborhood, choosing that which is closer in value. By this all edges are enhanced especially if a median is used before or after that. If applied before an edge detection very sharp edges are the result and in most cases thinning (see later) of these edges is not necessary. In the case of binary images the output image is equal to the input image. More than two iterations of ENHANCE give rise to homogenization and aggregation of regions.
Edge detection is realized by the gray level gradient (first spatial derivative) of the image function. The computation of its absolute value and direction is digitally approximated and generally the result of a convolution (see for instance the Sobel-Feldman-function). A further simplification is the so called **two-pixel**-gradient and defined by

$$\text{GRADIF}(\{G(I)\}) := \text{MAX}_{K=0}^{K=7}(\text{ABS}(G(ZP) - G(K)))$$

i. e. the maximum of the eight positive gray level differences concerning the value of the central point of the processing window. The output of the function, the edge strength, is the absolute value of the difference of

$$\begin{aligned} \text{GRADIF}(\{G(I)\}) &:= G(ZP) - \text{MIN}_{K=0}^{K=7} G(K) \text{ which resembles} \\ &:= G(ZP) - \text{RANK}^{9}_{1}(\{G(I)\}) \end{aligned}$$

In other words the image equals the original image which the eroded one is subtracted from (which is a simple example of the image algebra). The high sensitivity of this function with respect to noise is insignificant, if the original image is filtered before by MEDIAN. It is experimentally proved that combining MEDIAN and GRADIF has at least the same edge detecting properties as the well known Sobel-Feldman-convolution. Another approximation of the gradient may be defined by

$$\text{GRAD}(\{G(I)\}) := \text{RANK}^{9}_{8}(\{G(I)\}) - \text{RANK}^{9}_{1}(\{G(I)\})$$

which is similar, sometimes equal to the GRADIF with the advantage of internal smoothing by the MEDIAN. Here an eroded image is

substracted from a median-filtered one.

Generally speaking, the differences of two rank-procedures

$$RANK_{KL}(\{G(I)\}) := RANK^{9}_{K}(\{G(I)\}) - RANK^{9}_{L}(\{G(I)\}) \quad K < L$$

are extensions of rank functions and called range funktions (Bailey et al. 1985). They have been found to be useful for detecting edges and less sensitive to noise as explained above, and they all are GIPP-implementable.

4.2 Mask functions

The other type of local functions, implementable by the GIPP, is given by the mask functions. Using them the gray value of the central pixel can be changed, if and only if the gray level configuration within the window fits the claimed configuration of a mask from a given set. The conditions of a mask fitting may be described by logical functions, if binary images are used (to say template matching). But in the general case of gray scaled images, these conditions must be rewritten in a fuzzy manner by arithmetical inequalities.

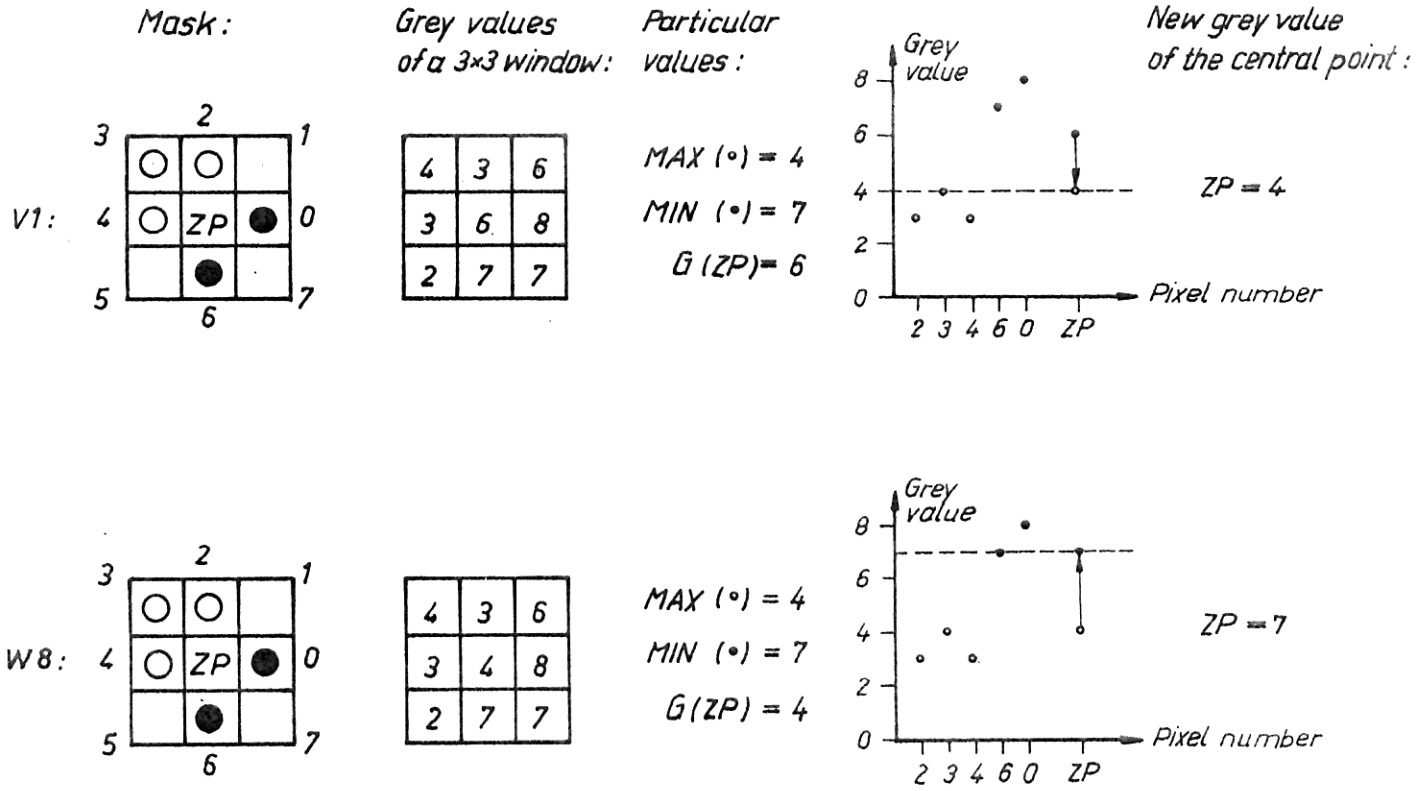


Fig. 4.2: Mask operations explained with 3*3 masks V1 (for thinning) and W8 (for thickening); for details see Roesler et al. 1980, 85

Three fundamental image procedures for object manipulations require mask operations: thinning, thickening and restoring (Roesler et al. 1980, 85). Each procedure is realized by several passes and in every pass several masks are to be applied. With thinning the gray level of the border lines of an object is decreased, with thickening it is increased, and restoring is a

mixture of both.
For selecting the new gray value of the centre-pixel in the processing window the following steps must be realized (for illustration see Fig. 4.2):

- firstly, the MAX(O) and the MIN(●) of the gray levels of all those pixels are computed which are marked by O and ● respectively (see for instance V1 or W8);
- next, the fit of the gray level configuration with one of the masks of the determined pass of the used function is proved by the validity of an adapted inequality (see below). It demands the gray level value of ZP to be placed between the two extrema (see on the right hand);
- eventually, if there is a fit, the value is altered by equating G(ZP) to one of the extreme values (see dashed lines), otherwise the value remains unchanged.

For decreasing the gray value of ZP a fit is indicated if the inequality

$$MAX(O) < G(ZP) \leqslant MIN(●)$$

holds, and the gray level of the centre-pixel is reduced to

$$G(ZP) := MAX(O)$$ (see top right).

For increasing the gray value of ZP, the inequality

$$MAX(O) \leqslant G(ZP) < MIN(●)$$

must be satisfied and the result is G(ZP) = MIN(●) (see bottom right).
After changing the value of ZP the fit is cancelled because the inequality can be satisfied no longer at that position of the processing window. The results $\{G_J(ZP)\}$ of all the masks of a pass are summarized by $MIN(\{G_J(ZP)\})$ and $MAX(\{G_J(ZP)\})$, respectively if a decrease or an increase had been indicated.
Thinning and thickening are homotopic processes by which the topology of the images is preserved; the topology may be changed only by restoring (for illustration see Fig. 4.3).

4.3 Freeman and node coding

Another type of mask functions can be used as a coding function to transform digital curves into a Freeman chain description.
Contours can be represented by Freeman-chains. A chain element leads from the center of one pixel to the center of the neighboring one. Freeman coding of contours may be done in a scanline manner, where to each pixel of the contour a code number is attached which depends on the 3x3-neighborhood. There are node elements and inner (continuing) elements. Inner elements are described by their Freeman code numbers. In the case of a node element the code number is supplemented by a code for the node state. Node states are beginning, ending, branching, connecting and crossing. To a beginning node the number of the column is attached. Each line is coded as follows:

LL = Line_beginning {node_state {column_number (if beginning node) {node_code}}|inner_code}

In fig. 4.5 examples are given for node and inner elements.

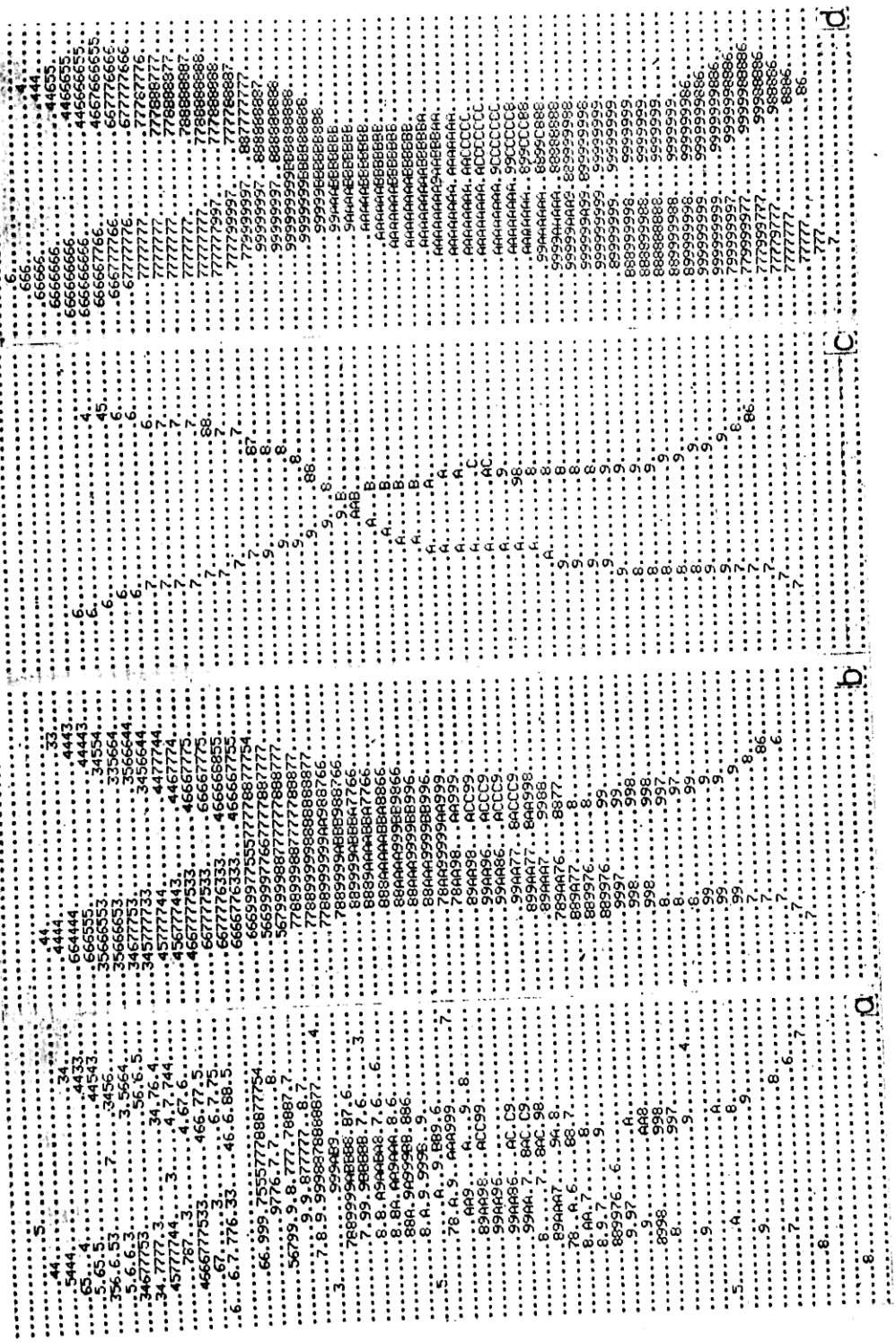


Fig. 4.3: Manipulation of a gray scaled model object
a: orginal, b: restored, c: skeletonized (by repeated thinning) d: thickened (by repeated thickening).

4.4 Image Algebra

A description of the methods of image algebra is given in (Serra, 1982) or in condensed versions in (Serra, 1986), (Sternberg, 1986) and (Haralick et. al. 1987).
The basic image algebra functions erosion, dilation, closing and opening can be implemented in the GIPP-structure. In this way also more complex functions, for instance convex hull estimation, medial axis transformation and so on, may be implemented.
Any sizes and shapes of the structuring elements can be realized:

1. Size of structuring elements:
   Structuring elements larger than 3x3 can be produced by a corresponding number of iteration steps. Fig. 4.6 shows a typical result for a 5x5-structuring element.
2. Structuring elements of any shape can be produced by the intersection operation of one or more dilation steps. (Bauernoeppel, 1984)

The second one of these cases can be realized either by a structure of parallely coupled GIPPs or by the use of two image memories and an interconnection network in which simple logical functions may be implemented.

4.5 Object labeling

Object labeling is one of the most important kinds of region oriented segmentation.
A useful algorithm was found by using of GIPP-structures. It processes an image in the following manner:
During a first, serial processing cycle of an image as described previously every beginning element (pixel) of the contained objects as well as every continuing element is labeled by a numeral. After this processing cycle all objects will be labeled by different numerals. One object may contain one or several numerals. During a small number of possibly following processing cycles the labeling of every object converges to the smallest contained numeral.
Convex objects are labeled unambiguously during the first processing cycle.

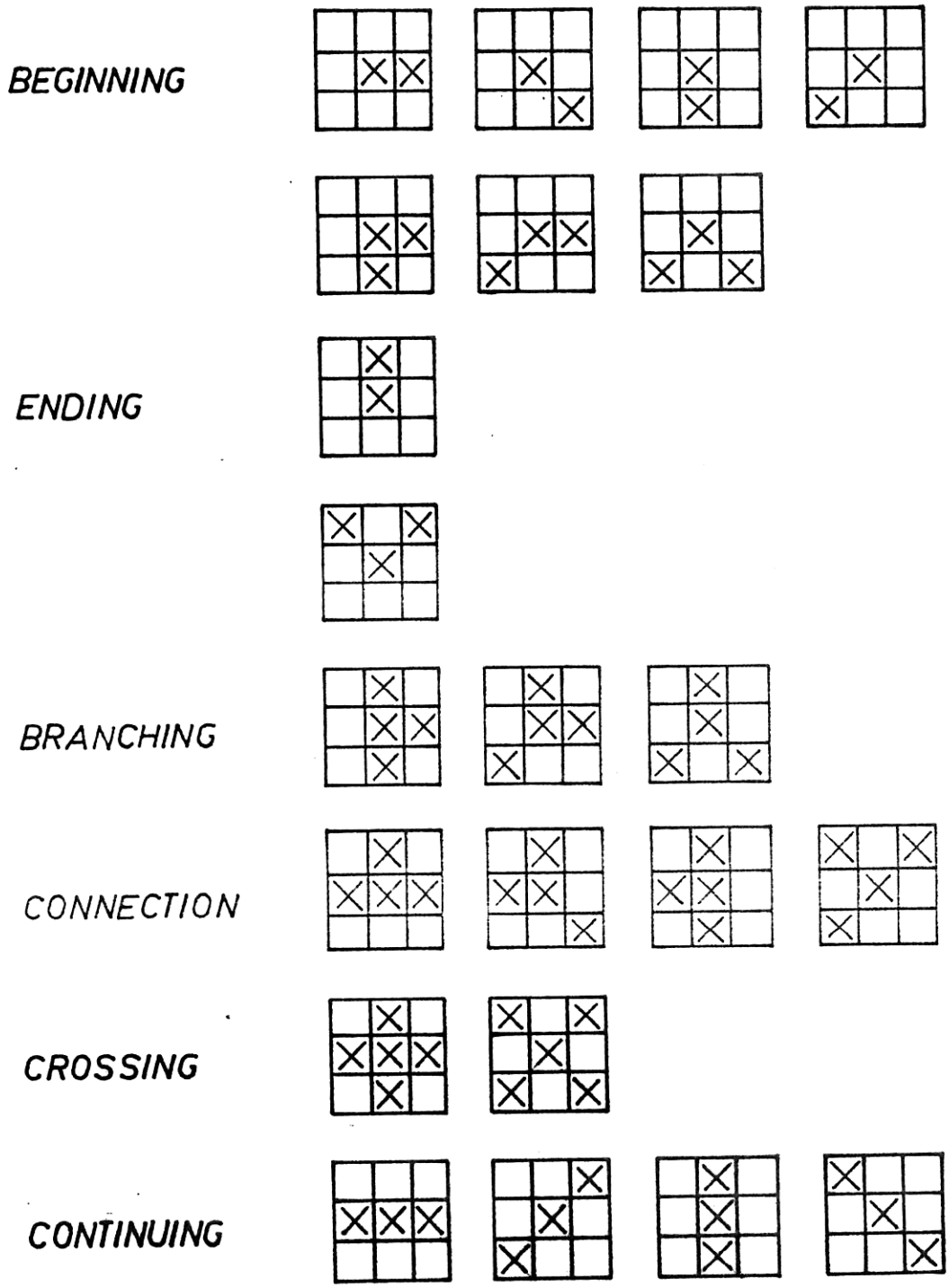


Fig. 4.5 Example for node and inner elements

Fig 4.7 shows an example for generating a crosslike-structuring element

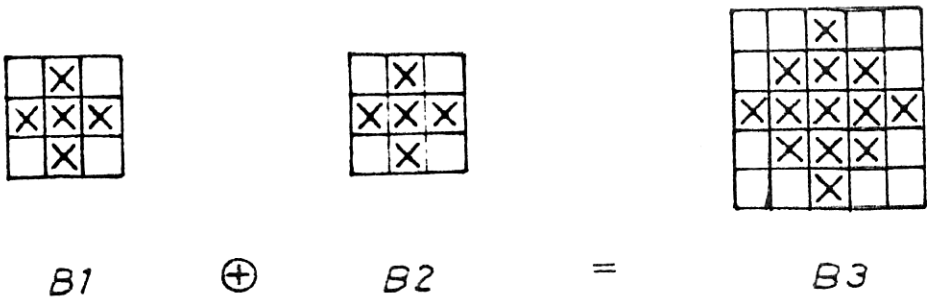


Fig. 4.6 Generating of masks larger than 3x3

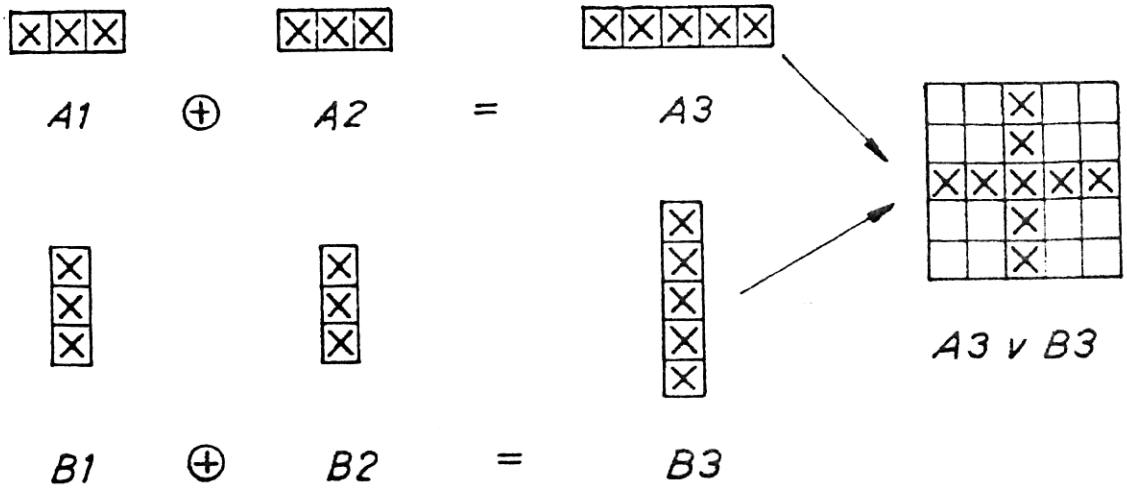


Fig. 4.7 Generating a mask by intersection

5. Implementation

5.1. Fundamental hardware structure of a GIPP

The realization of the described algorithm by means of a hardware structure is shown in fig. 5.1 and fig. 5.2. Especially because of the expected complexity of the register- and comparator-block and the needed RAM-size, the structure is designed for a 3x3 window only. Every number of gray levels, needed in practice (maximum 256 gray levels) may be realized.
The circuit scheme contains the following main parts:

- 3x3 register-block
  In the 3x3 register array the GIPP stores temporarily the gray levels of 3 consecutive pixels in corresponding positions of 3 consecutive lines [G(0)...G(8)]. Every input action moves the window one step along the lines of the picture, line by line.
- 3x3 comparator-block
  The nine gray values of the window are compared with the contents of register E g_m in every iteration step by the 3x3 comparator array. The output of the comparators are identically with the nine variables of the function F.
- two line buffers (delay lines)

The line buffers generate the input data of the 3x3 register-block in dependence of the serial image data input.

- table memory - 512 x 1 bit RAM
  The function F which determines the implemented image processing function is typically generated by a 512 x 1 bit RAM or RAM-segment.
- controller
  The controller has two functions:
  1. At the beginning of every iteration step the controller sets the register E on the half value of the available gray level range g^k_m.
  2. At the end of every iteration step the actual bit plane of register E is set or reset in dependence of the value of F.
- register E (result register)
  The register E is a buffer memory for the comparison-value g^k_m. At the end of computation, after k = m iteration steps, the result is found in register E.

If mask operations are carried out, one additional step k = 0 is needed. The controller sets the register E equal to the value of the centre-pixel. All gray levels of the window are compared with the gray level of the centre-pixel. If the mask condition (stored in the table-memory) is true, the new gray level will be selected by the following steps k = 1 to k = m. If the mask condition is not true, the register E keeps its value during all steps unchanged.

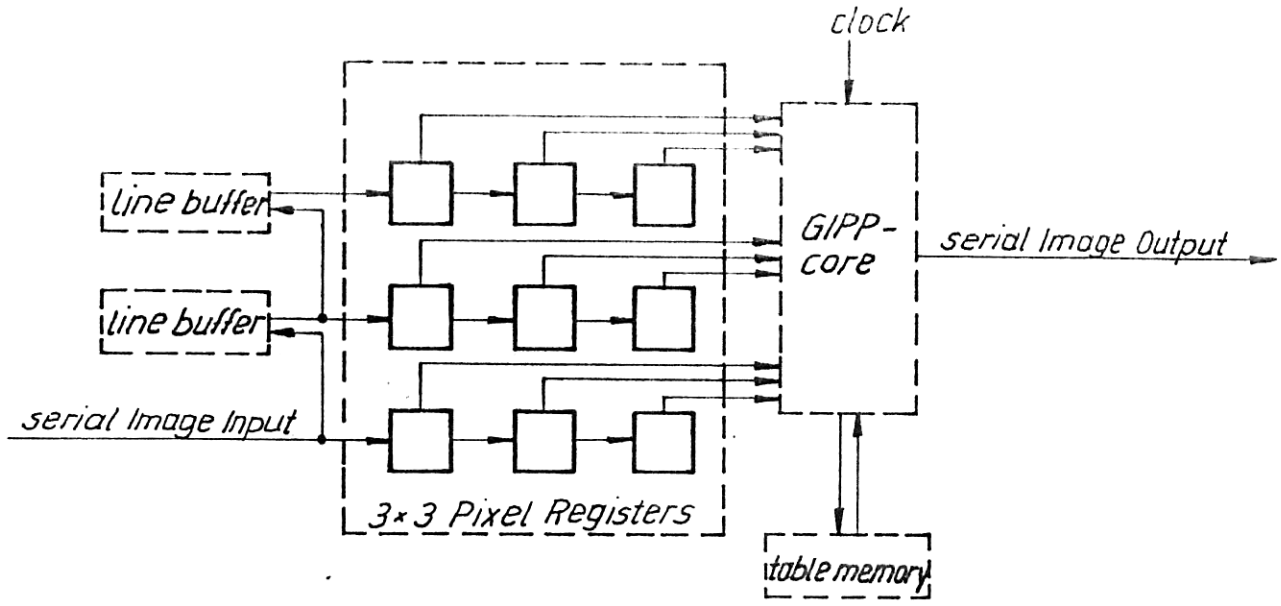


Fig.5.1 Block diagram of the processor GIPP

The GIPP-algorithm is characterized by four essential properties described previously:

- Images are processed by moving a 3 x 3 window in raster-scan manner.
- Within every bit plane the GIPP works in a parallel manner.
- The bit planes of a gray value are processed serially.
- The number of needed computation steps is constant and depends only on the given number of gray levels NG.

Because of these properties the GIPP is characterized by a high repitition rate of internal substructures and by a synchronous processing mode enabling the same processing time for every pixel.

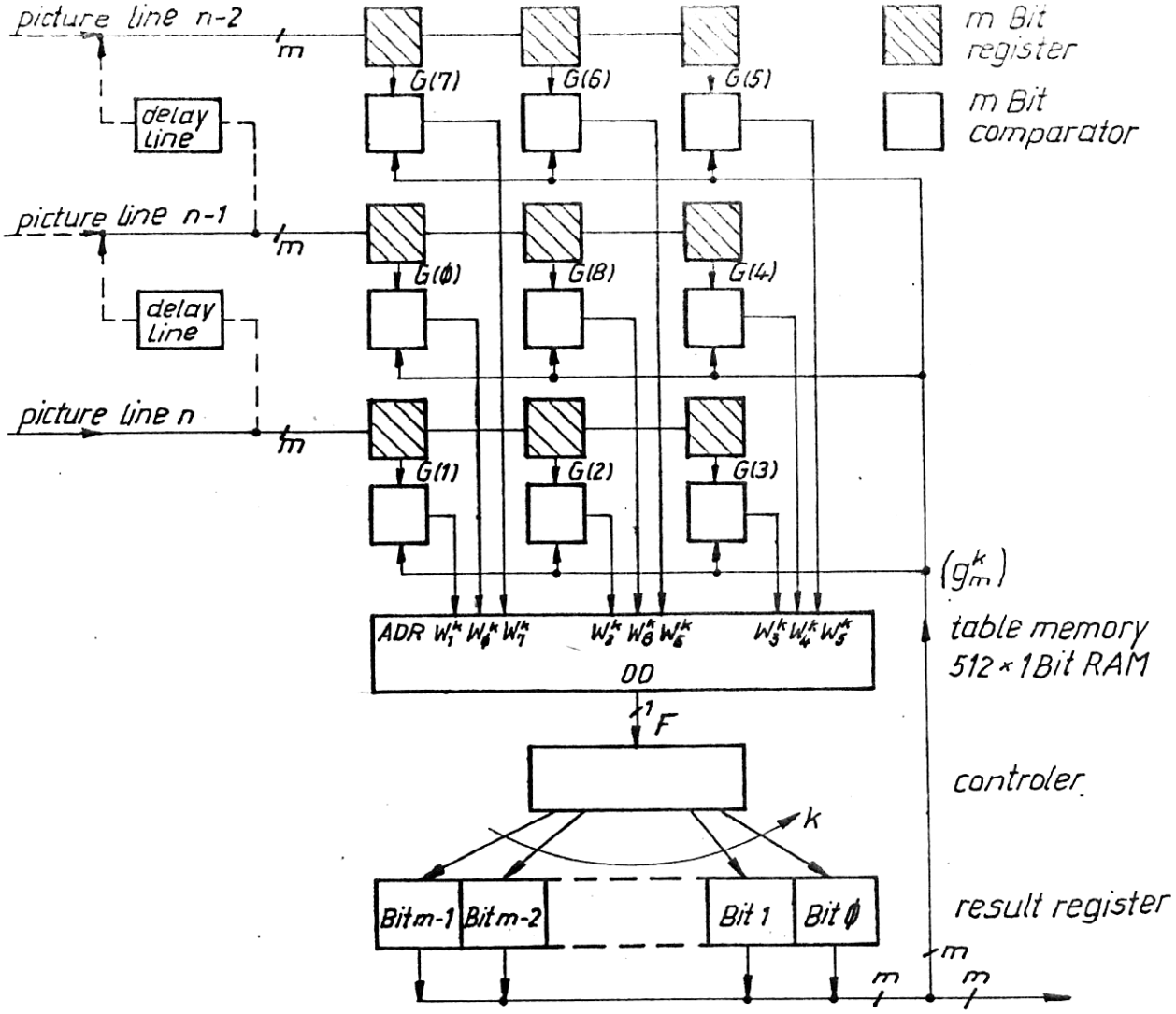


Fig. 5.2 Fundamental hardware structure of a GIPP

Further, a simple implementation of additional processing functions by a simple extension of the GIPP-hardware-structure, as described afterwards, is possible. In this way Freeman chain coding combined with node coding or other table-lookup operations can be realized (Nodes are pixels where chains are beginning or ending).
The described hardware-structure GIPP was realized by standard-IC's on a printed card and tested. The realized GIPP-boards are an integral part of the image processing system BAS 4000. The application of such systems is described elsewhere (Schwarze, 1987; Roesler et. al. 1988).

5.2 Semicustom chip realization of a GIPP

Fig 5.3. shows the block scheme of a GIPP prepared for semicustom chip realization. The main data lines, previously shown in fig.

5.2., are accentuated.
The 3x3 register-comparator-block is realized by the nine register/comparator units RCU0.....RCU8. The values of the corresponding pixels of three consecutive lines are transmitted to the register from the serial data input DI and the two line buffers. The nine gray level values of the window, stored in the window-registers, are compared with the contents of the result- and comparator controller RCC Vz (0...7). The nine outputs w_i of the comparators supply the address data A(0...8) of the table memory. The selected memory contents TS is transfered to the result and comparator-controller RCC which sets or resets the bits of the result register in the described manner. The actual contents of RCC Vz (0...7) becomes the new value of comparison for the next step. At the end of computation RCC contains the result which is transfered via E(0...7)to the output DA(0....7).
Besides of this basic operations the circuit includes a set of programmable processing functions and parameters allowing a high adaptability to given applications. The most important programmable features are:

- adaptable number of gray levels by a programmable length of the clock shift register CSR
- different operational modes, for instance programming modes for the table memory, mask and non-mask operation mode, borderline-detection mode, Freeman-code mode, test-modes a.s.o.
- different modes for GIPP-coupling: single-GIPP mode and pipe-line mode, high-speed coupling, special coupling for componant labeling a.s.o.
- programmable threshold for Freeman-code generation
- table memory selection
- indication of the actual line position and signalisation of the programmable line end for Freeman-code generation.
- The conditions of data output are programmable as well as they may be controled externaly.

List of used abbreviations (Fig. 5.1. - Fig. 5.4.)

data lines

DI	data input
DG	clock signal for data input
DA	data output (processed pictur data)
DO, ZEPU	data output (delay lines)
ZPOS	line position
CL1,2	clock signals
VZ	comparation value
W	comparator output
A	adress lines of the table memory
TS	selected RAM-contents
TSP	RAM-contents
ZPU	centralpoint
E	result

function blocks

DI/PC	data input/processing controller
AC	adress conter
LB1,2	line buffer
DIM	data input manager
RCU	register/comparator onit
PRU	programming register unit
TMC	table memory control
TM	tabele memory
RCC	result and comparator control
CSR	clock shift register
DOM/	data output manager/ subtraction unit

PRBD, SEL, WR, RD, ETU programming interface
DOA, ZPOA, ASI, EAS output control signals
SRP, UEE, UEA multiprocessor control
ZE1N, ZESN, ZG, RIN, TXEN, TXAN, CHEVN, ICTN, ZEI, ZAI, MINA, LK, UV, STX, T8, TYN, TY1N, TYBN, TYB1N, TY2N, TCL, TX1N
diverse clock, control and test signals

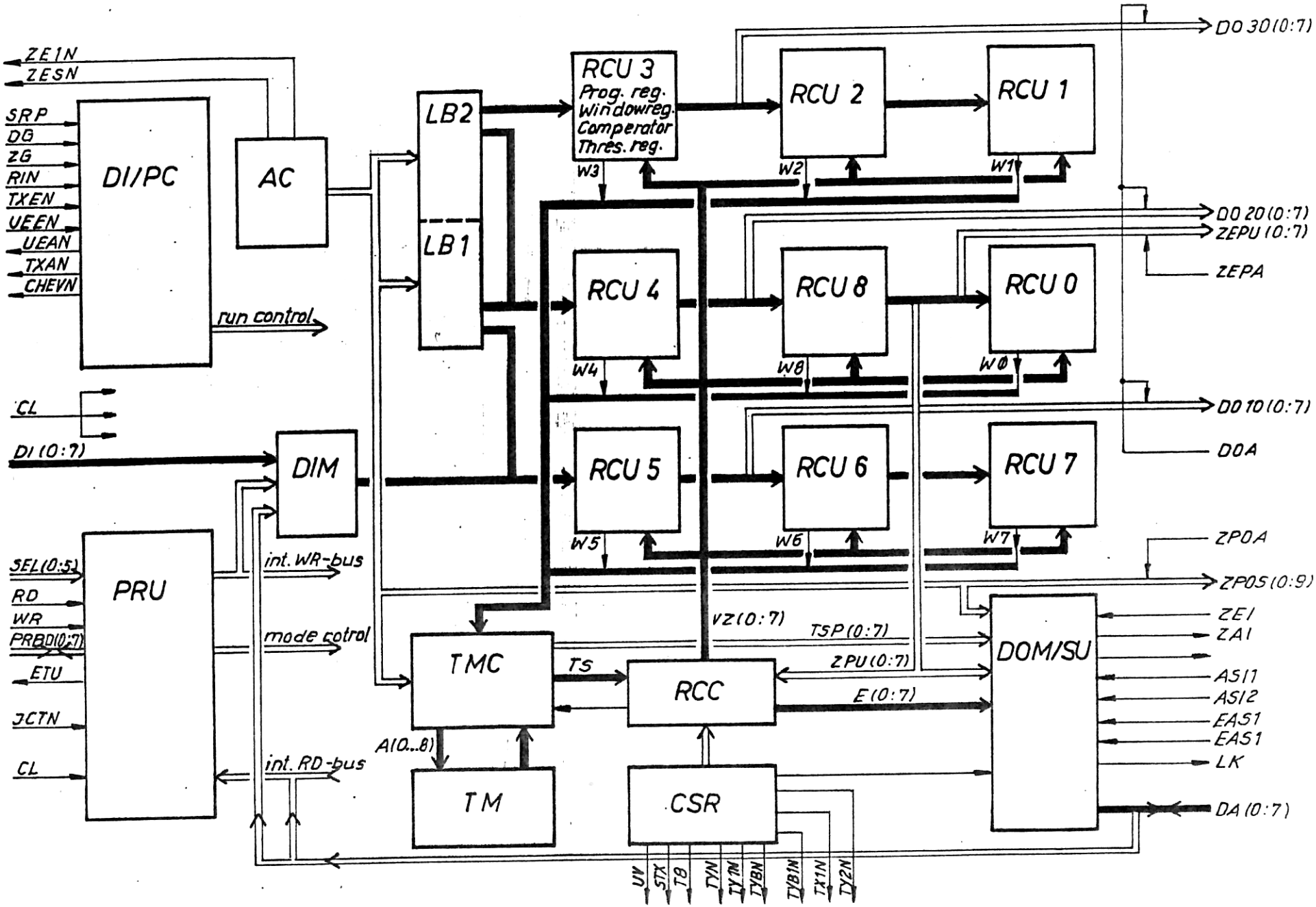


Fig. 5.3 Simplified block scheme of a GIPP, suited for realization as semicustom chip

The address-counter AC generates the line position ZPOS(0...9) (f.i. for Freeman-code generation) and the addresses for the line buffers.
The GIPP also may be used in multiprocessor applications. All needed control signals are generated and processed by the processor.
Implemented test functions allow a wide ranging test of the processor via the programming bus PRBD (0...7) even built in an application board.
The output lines of the 3 input-windowregisters of the 3x3 window are connected with pins of the circuit. This feature allows the use of the processor as fast sheft register with programmable length up to 2K.
All data outputs are realized by three-state drivers which may be controlled externally.
The computational speed in the single GIPP application is determined by the computation time of the bit planes, i.e. by the technologically based delay times of the GIPP especially by the access time of the table memory and line memories. Therefore the typical computation time of the proposed GIPP realization as semicustom chip is 100ns per bit plane. To compute one 8-bit pixel about 9 x 100ns = 0,9 us will be needed.

5.3 Possibilities of acceleration

Parallelization is the most important method to accelerate image processing procedures. Besides the GIPP-inherent parallel procedure additional methods of parallelization which are based on GIPP-arrays have been developed.

5.3.1. Pipelining of GIPPs

Pipelining of GIPPs is the practical realization of a programmable linear (1-D) systolic array.
Parallelization is realized by simultaneous processing of the pixels by several image processing functions available in different pipelined processing elements (PE).
Such a linear systolic array consisting of GIPPs is shown in fig 5.4.
The GIPP-array is characterized by some typical properties. Every I/O-operation is related to n computations. This property is desirable in practice, because usually the I/O-bandwidth of the processor or processor array and the data-source/sink as well as the computation speed of the PEs are restricted. In the described form of a GIPP array the maximum speed of a 8 bit-data stream is limited by the computation speed of the PEs to about. Both the common use of the data clock DC, which is transfered via the interface to all PEs, and the common processing clock signal CL guarantee a simple and reliable synchronisation. The constant processing time per pixel favours the easy use of the GIPP in systolic arrays independently of the programmed processing function. By the described linear systolic array consisting of n PEs n different image processing function may be performed simultaneously. The maximum processing speed corresponds to that speed of a single GIPP. Therefore the processing time needed to process an image is n-times shorter in comparison to a single GIPP application. It amounts about 60 ms to process a 256 x 256 x 8 bit image.

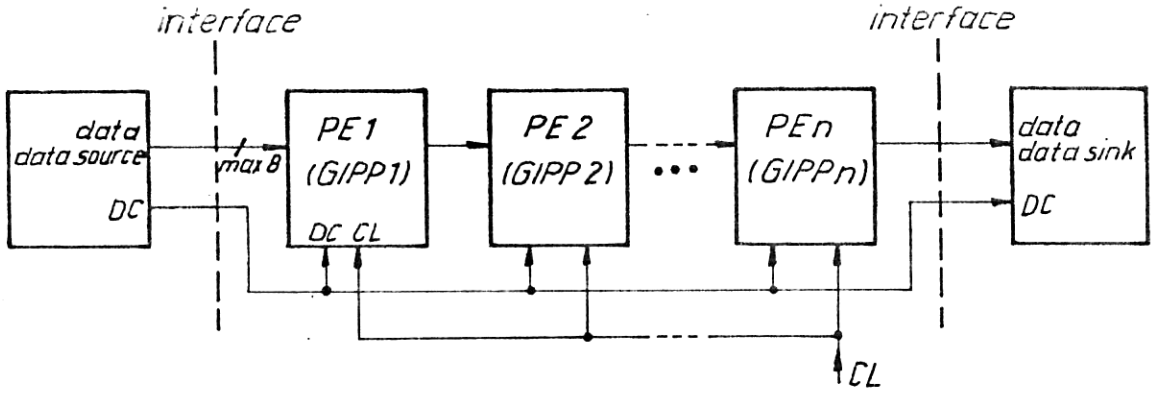


Fig. 5.4 Pipelining of n GIPPs

5.3.2. Fast processing circuit with cyclically coupled GIPPs

For high speed computation of image data a cyclic array of GIPPs was developed. The cyclic coupling of GIPPs (shown in fig. 5.5) is a special kind of massive parallelism.

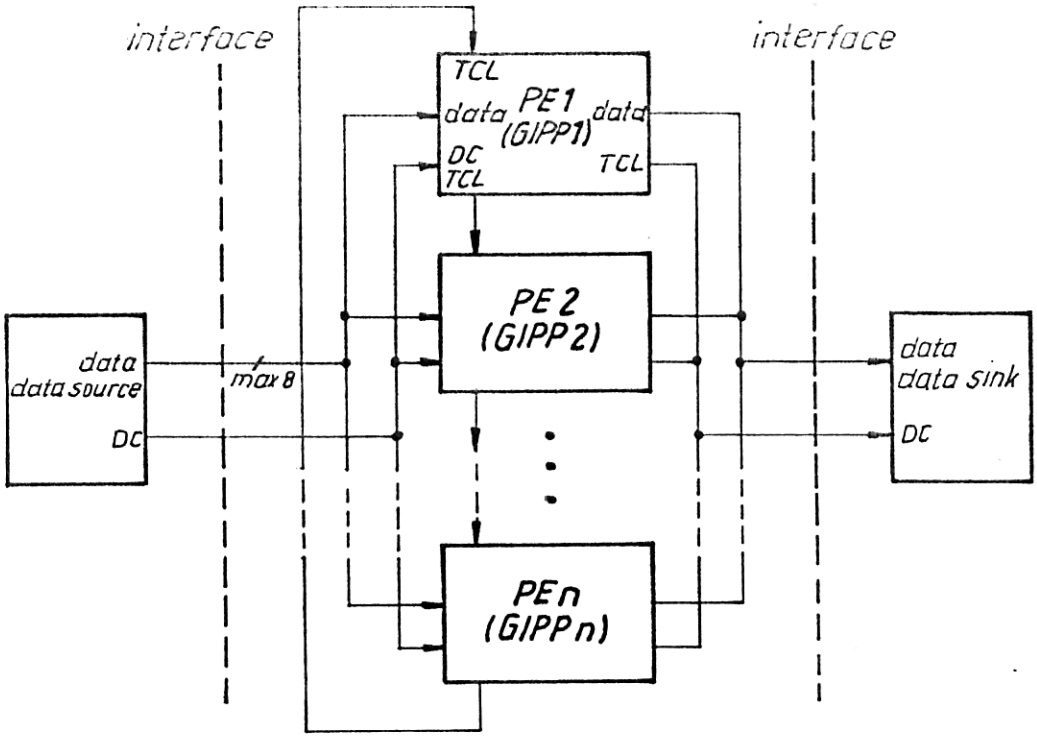


Fig. 5.5. Cyclic coupled GIPPs for fast processing

This massive parallelism is characterized by consecutive processing of several pixel groups from the whole image and parallel processing of all pixels which are elements of these groups. Usually an expensive and complicated data access to the data

source and sometimes also the data sink is needed to provide and to restore the image data. The cyclic GIPP array avoids this unfavourable property. It processes a serial, usually fast data stream. The array contains n GIPPs, called processing elements (PEs) in the following. These n PEs process n consecutive pixels of the image in parallel. Every pixel of the serial data stream releases its processing by the next PE of the cyclic array.
The array must contain such a number n of PEs, that each PE may become activated for a processing procedure only after closing of its preceding processing procedure. On the strength of the processing principle the maximum computation speed is determined above all by the permissible input/ output speed of image data, that is by the technological level. A data stream may be processed with a throughput of about 10 Mc/sec, which is typically for videorate.
The data input regime, needed in this mode, is programmable and controlled by the data input controler DI/PC (fig. 5.3).

5.4 Additional processing functions

5.4.1 Parallely coupled GIPPs

As described previously, contour oriented image segmentation as well as various procedures of image algebra may be realized by parallely coupled GIPPs. Fig. 5.6 shows the block scheme consisting of two parallely coupled GIPPs and an interconnection net work (arithmetic-logic-unit). The dynamic parameters of the circuit and the possibilities to build linear systolic arrays by pipelining of parallely coupled GIPPs (shown in fig. 5.6) correspond with the properties of a single GIPP.

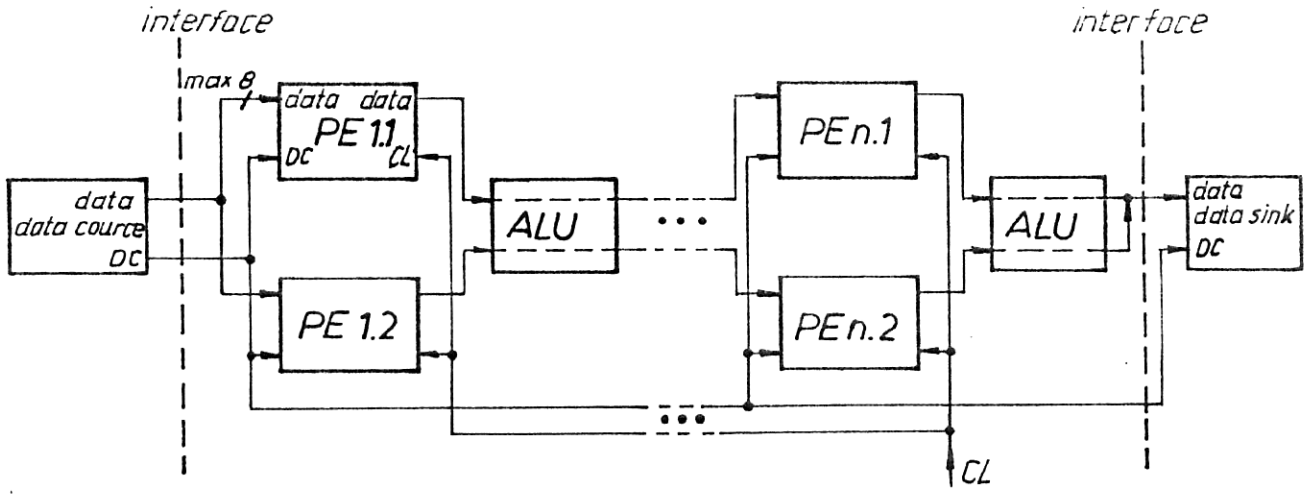


Fig. 5.6 Pipeline of parallely coupled GIPPs

5.4.2 Coupled GIPPs for region oriented segmentation

A hardware structure of two GIPPs based on the previously described algorithm was developed for tasks of region oriented segmentation. The dynamic parameters of the circuit correspond with the properties of a single GIPP.

During one or several processing cycles - the number of cycles is determined by the shape of the objects, contained in the image - all pixels of each object are labeled by the same numerals. The background is labeled by the maximum value of the available range of numerals.

6. Conclusion

The results in developing the GIPP processor are encouraging and are stimulating the VLSI implementation of GIPP processing elements.
With the aim to extend the parallelization of GIPP processing in an effective way systolic GIPP arrays have been developed. The systolic approach is very interesting in this relation providing a sound theoretical base for massively parallel GIPP processing.
Besides, this paper can also be considered as a contribution to the recent attempts to resume the neural network modelling research of the sixties and seventies. The GIPP processing structure was developed from first principles of local image preprocessing. In the authors opinion this approach is very promissing, mainly because of the facts, that the progammability of the processing elements is included in a natural manner and that the systolic approach may be used for developing massively parallel processing arrays.

7. References

Ataman, E. Aatre, V. K. and Wong, K. W.:
A fast method for red time median filtering.
IEEE Trans. Acoust. Speed, Signal Process.,
Vol 28 (1980), pp 415-420

Baisley D. G. and R. M. Hodgson:
Range filters: local intensity subrange filters and their properties.
Image and Vision Computing 3 (1985) 3, 99-110

Bauernoeppel, F.:
Implementierung von Algorithmen der Image-Algebra auf dem Spezialprozessor GIPP
Praktikumsbericht, ZKI, 1984

Chung, T. L.:
Entwicklung von Spezialprozessoren fuer die digitale Verarbeitung von Grauwertbildern
Dissertation A, ZKI Berlin, 1984

Duff, M. J.:
Computing Structures for Image Processing
Academic Press, London, 1983

Haralick, R. M. , S. R. Sternberg, X. Zhuang:
Image Analysis Using Mathematical Morphology
IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 4, 1987, 532-550

Hodgson, R. M. et. al.:
Properties, implementations and applications of rank filters.
Image and Vision Computing 3 (1985) 1, 3-14

Kempe, V. u. a.:
Bildverarbeitungssystem A 6472
rechentechnik/datenverarbeitung 19 (1982) 9, 8-11

Kramer H. P. and J. B. Brueckner:
Iteration of a Nonlinear Transformation for enhancement of Digital Images.
Pattern Recognition 7 (1975) 53-58

Kutschke, G. und G. Schwarze:
The programmable hardware-structure GIPP and its application
Mathematical Research Band 40, Akademie-Verlag Berlin 1987

Narenda, P. M.:
A Separable Median Filter for Image Noise Smoothing.
Proc. Pattern Rec. and Image Processing Conf. Chicago 1978, 137-141

Petkov, N.
Massive Parallelitaet durch ein systolisches GIPP-ARRAY.
16. Arbeitstagung Entwurf von Schaltsystemen und Systementwurf Dresden 1988
AdW/ZKI Berlin, April 1988

8. Supplement (edited 1/1990)

Further developements of the GIPP

Conception and technical parameters of the described GIPP-circuit are determined by the technological parameters of the used developing system. Because of the new political and economical situation in the GDR new high-level technologies are available now.
By the higher degree of integration and the increased dynamical parameters it is possible to develope the GIPP on the basis of a systolic principle (bit plane pipelining) (Fig.8.1) and arithmetical enlargements.

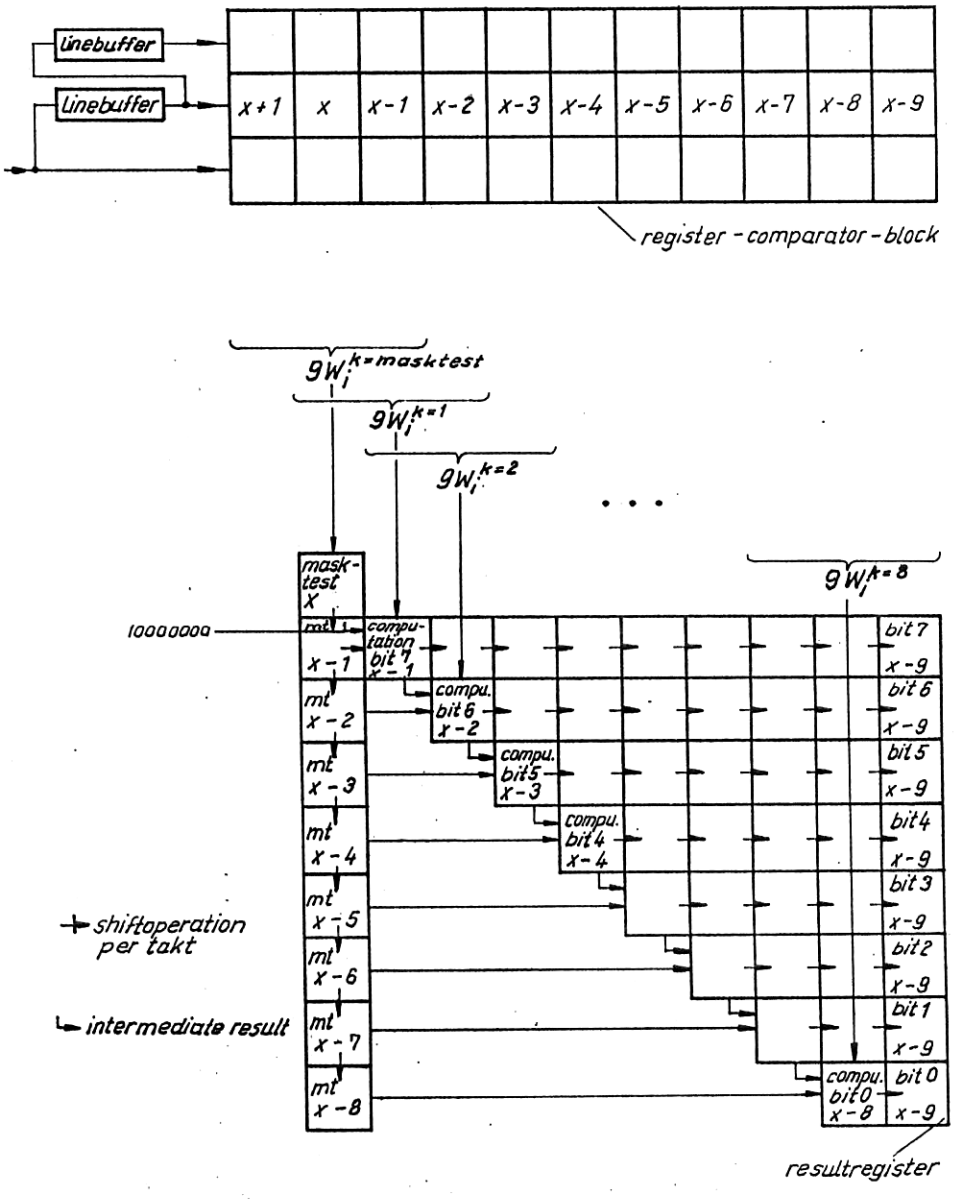


Fig. 8.1 Systolic processing principle

The most important advantage of this solution consists in the possibility to realize the parallelisation in the original circuit. No multi-circuit-arrangement is needed.
A set of deciding improvements may be realized:

*** The maximum of possible processing speed will be realized by one GIPP-circuit only. A further acceleration of picture processing procedures may be done by pipelining of GIPPs.

*** The maximum of processing speed may be increased up to 2Ø Mc/sec or higher (in dependence of the used technological base).

*** The possible number of bit-planes of a pixel may be increased from 8 up to f.i. 12 (8bit-pixel = 256 gray levels, 12bit-pixel = 4Ø96 gray levels).

*** Additional picture processing functions may be implemented in the GIPP, f.i. Laplace-, Sobel- and Prewitt-filter. By this way the application field of the GIPP may be enlarged from typically logical operations up to an important set of arithmetical ones.