An ASIC Performing Image Processing Tasks in Realtime

Reinhard Rauscher, Stefan Venzke [1]
University of Hamburg
Dept. of Computer Science
Hamburg, Germany
e-mail: rauscher@informatik.uni-hamburg.de

Abstract

A chip performing several common tasks in the field of image processing is presented. Most image processing tasks demand that the processing can be performed in realtime. For most of the operators realized this is only possible using dedicated hardware support. So, there has been the need for designing an ASIC performing a set of common and usually time consuming functions. The paper describes the most important design decisions, the results, the layout, and the aspects of testability.

1 Introduction

In the area of image processing [7] the analysis of pictures usually implies a set of preceeding, partially successive applied operations, e.g. thinning, smoothing, edge detection, and so on. This is at least valid for e.g. pattern recognition, character recognition, remote sensing, analysis of medical signals and images, roboter control, fingerprint and face identification, and target recognition [5, 17, 16]. Especially for these application areas the demand for realtime solutions increases [6, 13, 23].

Because of the pipelining opportunity of the components performing dedicated tasks - camera/scanner, analogue-digital converter, different components performing at least one dedicated task, and finally the analysis itself - this goal seems to be reachable, if each of these components fulfills the realtime condition. Especially for some common tasks there is the necessity for hardware performing this task as fast as possible. Furthermore, often there are other constraints which lead to an ASIC hardware solution, like power consumption, the size of the hardware and security aspects.

[1] Stefan Venzke is currently employed at MAZ GmbH, Hamburg

Although a lot of dedicated architectures exist especially developed for tasks in image processing (see e.g. [1, 5, 12, 11]) there is a need for specialized additional hardware to be used for the most time consuming tasks while other tasks will be handled by a common general purpose system. This approach leads to cheaper and even more flexible systems, which are fast enough to solve most image processing problems.

The intention of our hardware solution has been to decrease the calculation time by up to a factor of 100 or more [14] and to make a real pipelining concept for some most common tasks realisticly.

To avoid any misunderstanding, we did not design a general purpose image processor. An overview of such designs is given in [13]. Instead of this we decided to design a 'preprocessor' performing a set of usual, time-consuming tasks. As an application example, it could be integrated into scanners, cameras, and so on.

2 State of the art

Binary image processing is the area, in which most of the activities have been done. A lot of applications exists for binary images.

Additionally, they are used as interim results within the analysis of grey value images. Of course, there exists a lot of operations, which can be used for preprocessing tasks in image analysis. One of these preprocessing tasks is "thinning", mainly for this task dedicated hardware solutions have been published [14, 3].

3 Behavioral specification

The hardware presented in this paper has been developed with the objective to fulfill the following demands:

- It has to perform a set of the most common tasks in the field of binary image preprocessing.
- It should be integrated on a single chip.

- It must be cascadable (in a systolic manner). In such a cascade each component should perform one of the possible operations. The operation choosen must be dynamically changeable.
- It must be installable in numerous hardware environments, e.g. in a transputer environment as well as in a PC environment.
- It has to be faster than a commonly used camera itself. Therefore the data have to be processed and sent to the output at least as fast as the camera/scanner can produce the data.

The last item is explained in some more detail. Due to the demands for realtime applications there is the need for fast operating. Next, for avoiding the need for storing a huge part of the whole image, only a fixed static delay between input pixels and their corresponding output pixel is acceptable.

Regarding the large size of binary images (usually 512×512 or more) on one hand and the restricted size of on-chip memory on the other hand, it is impractical to store a whole image on the chip. Thus, our ASIC realizes only those types of operators, which are based on an analysis of local neighbourhood of the pixel under consideration. All operations are based on the well known frame technique, using a 3×3 picture frame, scanning the entire image. So, only a small area of the whole image has to be stored within the ASIC. This is shown in Figure 1. In the figure the different rows of the input image are sketched. Only the dashed area consisting of roughly three rows has to be stored in the ASIC. Additionally, a few pixel under consideration (in the same row) are sketched by the black area. For all these pixels the new value has to be determined in parallel.

Our design implements the following concept: The ASIC gets input data (binary pixel values) from a camera. The incoming pixel are stored in on-chip-RAMs as long as they are needed for calculating the output pixels.

Only such image processing operations have been realized, which are based on a $3\ x\ 3$ neighbourhood. The input and the output has to be organized row by row, as it is done in usual cameras.

The typical clock rate of video/CCD cameras is approximately 33 MHz. This clock rate is to fast for this approach. Therefore some pixel values have to be determined in parallel. The rate of this parallelism needed is given by:

$$par \cdot cl_{sys} \geq cl_{camera},$$

where par specifies the rate of parallelism, cl_{sys} the clock rate of the system and cl_{camera} the clock rate of the camera.

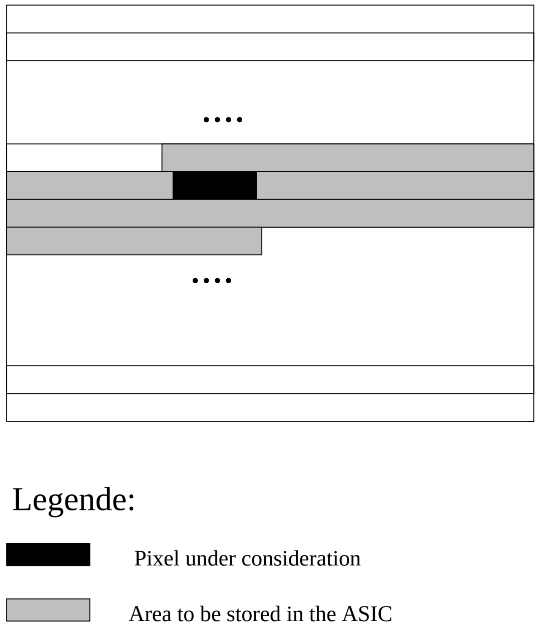

Figure 1: Illustration of storing only a small part of the image

We chose the value $par = 8$ for two reasons:

- Most often the RAMs are organized as 8 bit RAMs
- There exists a lot of hardware components supporting byte parallel data transfer (e.g. CO11-component [19]).

Based on the value $par = 8$ the demand for

$$cl_{sys} \geq 4.125 \text{ MHz results.}$$

This value for cl_{sys} has been easily reached using our approach with several on-chip-RAMs and using a low cost design system (e.g. SOLO 1400 [4]).

The parallelism as discussed above, leads to a restriction, namely only such operators can be realized in our approach, which can be performed in parallel.

Cascadability as depicted in Figure 2 implies that input and output data and their transfer have identical structure. So, it becomes possible to run several (optionally different) operators on one image in a pipelined manner. The time spent between starting the first operator and ending the last one is only a little bit longer than the time spent for only one operator. As sketched in Figure 3 it is possible to change the queue of operations dynamically. This can be done

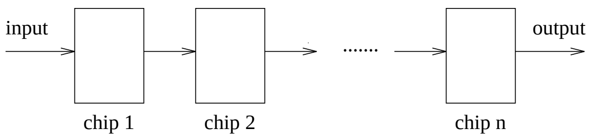

Figure 2: A cascade of (identical) ASICs

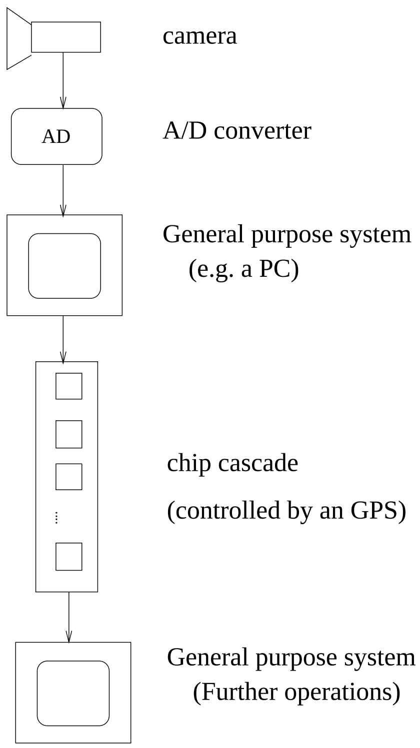

Figure 3: Hardware environment of the ASIC(s)

changing the control inputs of each of the ASICs within the chip cascade. Please note, that using the first general purpose system (as sketched in Figure 3) is optional. Instead of this, the chip cascade could be directly connected to the AD converter.

4 The realized operators

In the discussion of the operators implemented in our ASIC, we make use of the following basic definitions:

- Each pixel in a binary image has a binary value. It is either *foreground* (in this case it is part of an object within the analyzed image) or it is *background*.

- In the following we use the convention:
  - Pixels of foreground have the value "1".
  - Pixels of background have the value "0".

- In the literature it is distinguished between *4-connectivity* and *8-connectivity*.

  In *4-connectivity* 4 neighbour pixels are considered, while in *8-connectivity* all 8 direct neighbour pixels are considered. Both connectivities are sketched in Figure 4.

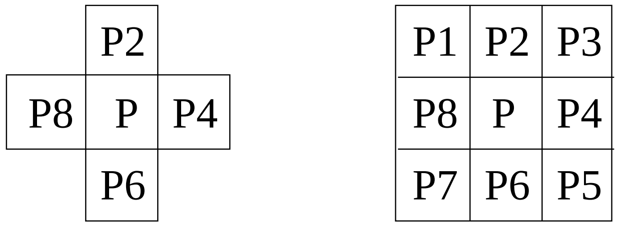

Figure 4: 4- and 8-connectivity

Now, the realized operators are discussed. As mentioned above, all of these operators can be executed in parallel. Keep in mind, that operators can be applied either on foreground pixel or on background pixel, alternatively.

The operators realized within the ASIC are:

0 : "Removement of isolated pixels" (noise elimination), based on 4-connectivity
1 : "Removement of isolated pixels" (noise elimination), based on 8-connectivity
2 : "Thinning", subiteration I
3 : "Thinning", subiteration II
4 : "Erosion", based on 4-connectivity
5 : "Erosion", based on 8-connectivity
6 : "Edge detection", based on 4-connectivity
7 : "Edge detection", based on 8-connectivity
8 : "Extended Erosion", based on 8-connectivity
9 : "Extended edge detection2", based on 8-connecctivity
10 : "Dilation", based on 4-connectivity
11 : "Dilation", based on 8-connectivity
12 : "Extended dilation", based on 8-connectivity
13: "Identity"

In the following the different operators are explained in detail. Only operators based on 8-connectivity are explained. The operators based on 4-connectivity are defined analogous.

4.1 Removement of isolated pixel

This operator is used for noise elimination. The new value of a pixel P depends on its current value and the values of its 8 neighbours. It is calculated according to:

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i > 0 \\ & \wedge \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

4.2 Thinning

The thinning operator is used within a huge field of applications. It generates successively the skeleton of a given object. An introduction in this operator and its application can be found in [18]. An overview and a comparison of different thinning algorithms is given in [8, 9, 20, 2, 15, 10, 21].

In our realization we chose the algorithm of Guo and Hall [8]. This algorithm generates "good" skeletons on one hand. On the other hand it works based on a 3×3 neighbourhood. The disadvantage is, that two different subiteration steps have to be implemented. A realization of this thinning algorithm uses a combinatorial circuit containing approx. 150 gate equivalents for each pixel.

4.3 Erosion

The operator "Erosion" is used in a similiar way as the thinning operator. It realizes the following function:

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i = 8 \\ & \wedge \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

4.4 Edge detection

The operator "Edge detection" is used for detection of edges of an object. Only those pixel remain the value '1' which are part of the edge. The operator can be described as follows:

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i < 8 \\ & \wedge \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

4.5 Extended erosion

The usual "Erosion" operator as described above is very susceptible against noise. For improvement of the susceptibility the "Extended erosion" has been defined.

It has the following form:

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i \geq 7 \\ & \wedge \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

4.6 Extended edge detection

This operator has been defined to improve (analogue to "Extended erosion") robustness of edge detection in the presence of noise.

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i < 7 \\ & \wedge \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

4.7 Dilation

The "Dilation" operator determines, which background pixels are neighboured by foreground pixels. Each of these pixels is converted to a foreground pixel.

"Dilation":

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i > 0 \\ & \vee \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

4.8 Identity

The "Identity" operator is used for testability aspects. For testing a pipeline of ASICs it is helpful to have the opportunity to compare directly the output image against the input image. In addition, the "Identity" operator is used in case, that the pipeline contains more ASICs than are needed for the operations to be currently performed.

4.9 Optimizations

Obviously, all of the operators are realized using a combinatorial circuit. Additionally, parts of the circuitry can be shared between different operators. This will be explained using the exampe of "Erosion" and "Dilation". Due to the opportunity of switching between foreground and background it can be seen very easily, that Erosion of foreground will result in the same image as dilation of background.

"Erosion":

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i = 8 \\ & \wedge \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

"Dilation":

$$P_{new} = \begin{cases} 1, & if \quad \sum_{i=1}^{8} P_i > 0 \\ & \quad \vee \quad value(P) = 1 \\ 0, & otherwise \end{cases}$$

$\Leftrightarrow$

$$\begin{cases} neg(1), & if \quad \sum_{i=1}^{8} neg(P_i) = 8 \\ & \quad \wedge \quad neg(value(P) = 1) \\ neg(0), & otherwise \end{cases}$$

In addition, some operators use identical boolean subexpressions. An example for this are operators, which are distinguished only by different connectivities, e.g. "Erosion based on 4-connectivity" and "Erosion based on 8-connectivity". Of course, the function to be determined using 4-connectivity is a subfunction of that used for 8-connectivity.

Finally, two different functions concerning to different operators can be meshed. As an example Figure 5 demonstrates the meshing of the "Erosion" operator with "Edge detection".

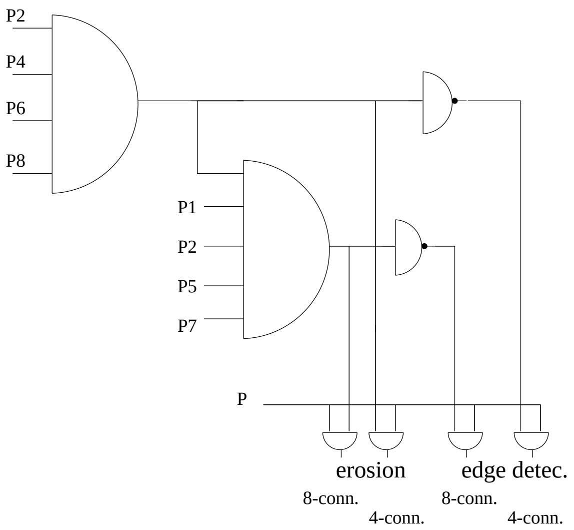

Figure 5: Meshing "Erosion" and "Edge detection"

5 Architecture

5.1 Overview and design constraints

Additional to the demands described in Chapter 2 the following demands had to be fulfilled in the design of the ASIC:

- The number of pixels within one line of the image should be adjustable between 512 and 1024.
- The data transfer between ASICs has to be realized by handshake.
- Problems with the edges of the input image have to be considered.
- Images should be analyzed in serial with no or only a few wait cycles between two images.
- There should exist an additional output, which indicates, whether the actual operator made a modification of the actual image. This output is needed especially for the thinning operator.
- There should be a opportunity to perform an incircuit test of the ASICs.

Figure 6 shows an overview of the architecture. It consists of four mayor parts, named *preparation unit, calculation unit, modification control unit,* and the *control unit.* These subunits are presented in the following sections.

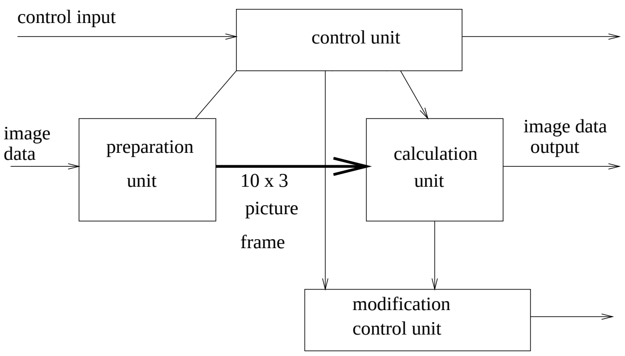

Figure 6: Overview of the architecture

5.2 Preparation unit

This unit stores the input data as long as they are needed for performing the current operation. Since the calculation unit needs a 10 $\times$ 3 pixel frame for an operator, the preparation unit has to store the last three rows of the image. This frame contains the neighbourhood of eight pixel under consideration.

The schedule of the preparation unit is as follows: First, the input data for eight pixel are stored in the input buffer. They are transferred to the on-chip-RAMs and to the pixel frame buffer. In addition, the remaining pixel values of the 10 $\times$ 3 frame are determined from the last two lines of the actual image. In Figure 7 the most important parts of the preparation unit are sketched.

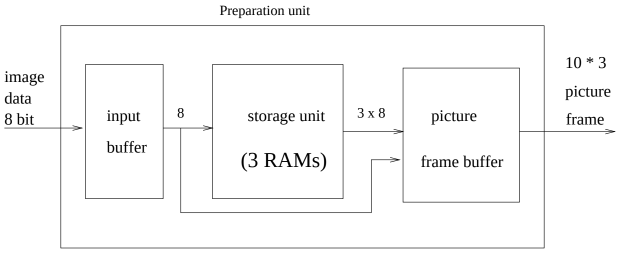

Figure 7: Structure of the preparation unit

5.3 Calculation unit

This unit has the task to determine 8 new pixel values dependent on the choosen operator. This is done using the 10×3 pixel frame coming from the preparation unit. In Figure 8 the basic structure of the calculation unit is illustrated. For each of the eight pixel for which a new value has to be determined first a 3×3 neighbourhood is generated. For each of the pixels the new value is determined using a combinational circuit. Please note, that the whole calculation unit could be assumed to be a combinational circuit which evaluates the new values of the 8 pixel under consideration in parallel.

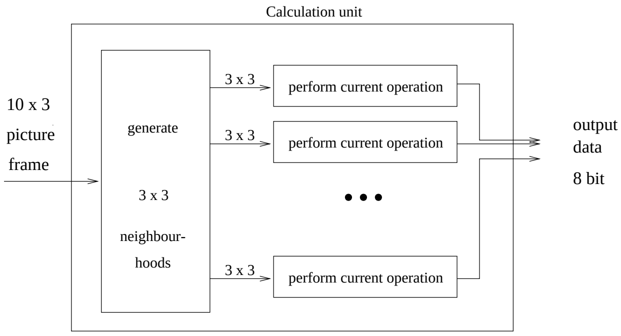

Figure 8: Structure of the calculation unit

5.4 Modification control unit

This unit determines, whether the new image and the input image differ in some pixels.

It is used additionally to determine how many pixel values have been changed in the actual image.

These values could be necessary for the decision, whether to modify the resulting image by using the chip cascade once more or to stop the preprocessing activities. As an example for this, the *thinning operator* can be considered.

While thinning results in modification of the input image, the thinning procedure is not finished.

6 Results

The ASIC has been designed [22] using the SOLO 1400 system [4] and is realized in a 1.5 μ CMOS process. The chip area is given in the following:

Array area:	$4.69 \times 3.94 = 18.49$ mm^2
Active chip area:	$5.86 \times 5.12 = 30.00$ mm^2
Die size:	$6.07 \times 5.33 = 32.35$ mm^2

The resulting layout, with the 3 RAMs and additionally two blocks of standard cells as mentioned above, is shown in Figure 9. For further information (e.g. pads, pinout) we have to refer to [22].

The maximal clock frequency is at least 16.66 MHz. This has been determined as a result of several simulation runs. Since exactly two clock cycles are needed for each input, the input data rate can be estimated to be at most 8.33 MHz. So, a modification of an image sized 1024×1024 pixels using one or more ASICs would take 0.03 seconds (independent from the operator performed). In comparison to the demands remarked in the behavioral specification the ASIC could run a factor of two faster than necessary in today's applications.

7 Testability aspects

For testability aspects we made some analysis concerning the possible faults and their relevance for the following image analyzing procedure. Therefore we classified the possible fabrication faults as follows:

1. Faults, which do not change the data output.
2. Faults, which may be relevant on the data output, but would have no more effects than noisy data. Because of the cascadability of the ASICs this kind of faults can be quite harmless.
3. Faults, which are of crucial relevance (e.g. problems with power supply channels, *stuck-at* at the output lines).

The most interesting result has been, that faults of the third kind are less than 1 % of all possible faults. On the other side, most of the faults of the second kind could be quite easily detected. This becomes clear considering test images (e.g. pattern like a chess board). Using these images most of the faults within the RAMs and within the circuits for performing the operators can be detected. So, a strategy has been developed to perform an in-circuit test of the ASICs.

Additionally, we implemented a usual scan path and build-in-self-tests (BIST) for testing the combinatorial circuits and the RAMs, respectively.

Figure 9: Layout of the chip

8 Summary/further activities

An ASIC has been realized which performs a set of the most common low-level operators in image processing. A sequence of operations can be applied on a continious video data stream.

In the near future, the functionality of the operators and the real application of the ASIC will be evaluated using practical examples for different applications.

References

[1] A. Basu, C. Brown: Algorithms and Hardware for Efficient Image Smoothing, Report TR 149, University of Rochester, N.Y., 1984

[2] Y.S. Chen, W.H. Hsu: A Comparison of some One-pass Parallel Thinnings, Pattern Recognition Letters, 11, pp. 35-41, January 1990

[3] C.S. Chen, W.H. Tsai: A New Fast One-pass Thinning Algorithm and its Parallel Hardware Implementation, Pattern Recognition Letters, 11, pp. 471-477, July 1990

[4] European Silicon Structures Ltd., Solo 1400 Release 3.1 Reference Manual, Berkshire, 1991

[5] K. Fu: VLSI for Pattern Recognition and Image Processing, Springer Verlag, Berlin, 1984

[6] C.J. Georgiou, D. Anastassiou: A Real-Time, Single Chip, Laplacian Image Edge Detector, Research Report IBM, Yorktown Heights, RC 10549, 1984

[7] R. Gonzalez, P. Wintz: Digital Image Processing, Reading, Mass., Addison Wesley, 1977

[8] Z. Guo, R.W. Hall: Parallel Thinning with Two-Subiteration Algorithms, Communications of the ACM, Volume 32, Number 3, pp. 359-373, March 1989

[9] R.W. Hall: Fast Parallel Thinning Algorithms: Parallel Speed and Connectivity Preservation, Communications of the ACM, Volume 32, No 1, pp. 124-131, Jannuary 1989

[10] C.J. Hilditch: Comparison of Thinning Algorithms on a Parallel Processor, image and vision computing, Volume 1, Number 3, pp. 115-132, August 1983

[11] J.Kittler, M. Duff: Image Processing System Architectures, Research Studies Press, Great Britain, 1985

[12] S. Levialdi: Integrated Technology for Parallel Image Processing, Academic Press, Orlando, Florida, 1985

[13] H. Li: VLSI Architecture for Image Processing, Research Report IBM, Yorktown Heights, RC 10816, 1984

[14] A. Maeder, R. Rauscher: A Chip for Realtime Skeleting of Images, Proceeding Custom Integrated Circuit Conf., Boston, MA, 13.-16. May 1990, No. 15.4

[15] N.J. Naccache, R. Shinghal: SPTA: A Proposed Algorithm for Thinning Binary Patterns, IEEE Transactions on Systems, Man and Cybernetics, Vol SMC-14, No 3, pp. 409-418, June 1984

[16] M. Onoe, K. Preston, A. Rosenfeld: Real-Time/Parallel Computing, Image Analysis, Plenum Press, New York, 1981

[17] M. Onoe, K. Preston, A. Rosenfeld: Real-Time Medical Image Processing, Plenum Press, New York, 1980

[18] A. Rosenfeld, A. Kak: Digital Picture Processing, Vol. 2, Academic Press, New York, 1982

[19] M. Rygol, T. Watson: Connecting Links, INMOS Technical Note, 1988

[20] R. Smith: Computer Processings of Line Images: A Survey, Pattern recognition, Volume 20, No 1, pp. 7-15, 1987

[21] H. Tamura: A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint, Proceeding of the International Conference on Pattern Recognition, IEEE, New York, 1978, pp. 715-719

[22] S. Venzke: Analyse von Bildverarbeitungsaufgaben für Binärbilder und deren auf Fenstertechnik basierende Implementierung in einem VLSI-gerechten Architekturkonzept, Diploma thesis, University of Hamburg, 1993

[23] F. Wahl: A Binary Image Preprocessor for Document Quality Improvement and Data Reduction, IBM, Th. Watson Research Center, Yorktown Heights, RZ 1438, 52141, 1985