Application research of an optimized pixel template image correlator

van Oosterhout, P.H.J.

Award date:
1992

EINDHOVEN UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
Measurement and Control Section

Application research of an

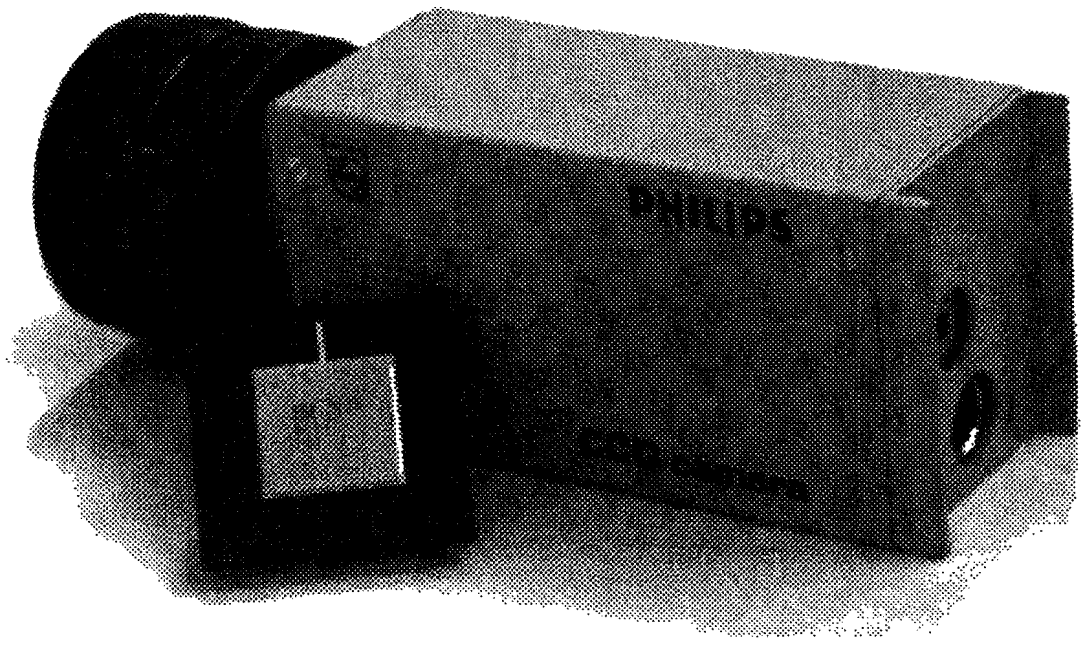
OPTIMIZED PIXEL TEMPLATE IMAGE CORRELATOR

by P.H.J. van Oosterhout

M.Sc. Thesis on practical training period
carried out from october 1991 to august 1992
commisioned by prof. dr. ir. A.C.P.M. Backx
under supervision of ir. N.G.M. Kouwenberg
date: 19 august 1992

Application research of an

Optimized Pixel Template Image Correlator



author :	P.H.J. van Oosterhout
id.nr. :	236356
institute :	Technical University Eindhoven
department :	Electrotechnical Engineering
professor :	prof. dr. ir. A.C.P.M. Backx
coach :	ir. N.G.M. Kouwenberg
company :	Nederlandse Philips Bedrijven BV
department :	CFT - ISP - IMI
coach :	ing. A.J.M. van Lier
period :	october 1991 - july 1992

ACKNOWLEDGEMENT

At this point I would like to thank all the people who were of great help during my graduation period at Philips CFT.

First of all I would like to thank ir. R.G. van Vliet, who arranged the graduate work at Philips CFT. Of course I also want to thank ir. N.G.M. Kouwenberg, who toke over the work from mr. van Vliet and supported me during my final assignment.

Working under ing. A.J.M. van Lier at Philips was a new experience. Within the required demands, he gave me complete freedom in my research. If problems occurred, he was always willing to help. Thanks to this approach I learned a lot about image processing techniques.

Last but not least I would like to thank all my colleague students at Philips CFT.

Joeri van Oosterhout
August 1992

SUMMARY

Fulfilling a final assignment is necessary to graduate at the Technical University Eindhoven, Department of Electrical Engineering. This report is the end product of the assignment performed at the Philips Centre for Manufacturing Technology. It is an investigation into the application possibilities of an Optimized Pixel Template Image Correlator (OPTIC). This custom chip implements different kinds of picture processing techniques which are based on greylevel images. The chip performs these techniques real time with a speed up to 20 Mega-pixels per second.

After building and testing an OPTIC testcard which contains the necessary circuits to drive two OPTICs the application research started. During this research a test program and a demo program has been written. With these two programs it is possible to demonstrate the applications of the pixel processing chip.

Thanks to the performed research it has become clear that the OPTIC can be used for operations which are necessary to perform some types of filter and recognition techniques.

The following on morphological operations based applications possibilities are found for the OPTIC:

- Erosion and dilation
- Opening and Closing
- Noise filtering
- Edge preserving filtering
- Dynamic thresholding
- Edge detection
- Greylevel template matching
- Template matching with Non Maximum Suppression

As a result of the fact that the OPTIC uses greyvalue Max-Min filtering techniques the operations are more or less independent of the amount of the available light and the focusing of the camera.

CONTENTS

LIST OF ABBREVIATIONS

CFT	Centre for Manufacturing Technology
DC	Don't Care pixel attribute
DRC	Dynamic Range Correlator
ESPRIT	European Strategic Programme for Research and development in Information Technology
FIL	Filter pixel attribute
IMI	Industrial Measurement and Inspection
ISP	Industrial Signal Processing
LSB	Last Significant Bit
MAX	Maximum pixel attribute
MIN	Minimum pixel attribute
MSB	Most Significant Bit
MUX	Multiplexer
NMS	Non Maximum Suppression
OPTIC	Optimized Pixel Template Image Correlator
PAPS	Picture Acquisition and Processing System
PEC	PAPS Experimental Card
RAM	Random Access Memory
SBIP	Single Board Image Processor

1 INTRODUCTION

Fulfilling a final assignment is necessary to graduate at the Technical University Eindhoven, Department of Electrical Engineering. This report is the end product of such an assignment done at Philips Centre for Manufacturing Technology (CFT). The assignment is performed at the group Industrial Signal Processing, Industrial Measurement and Inspection (ISP-IMI).

In the electronics industry severe inspection problems are generated by the combination of the increasing miniaturisation of electronic components and assemblies, the increasing speed of their production, and ever-increasing demands for high quality. For that reason the demand for inspection systems with high accuracy raised. The key objective of the ESPRIT project #2017 is the development of such inspection systems.

In co-operation with Silicon and Software System, Philips has developed a custom chip as part of the TRIOS project. This chip implements different kind of picture processing techniques which are based on greylevel images. The chip can perform these techniques 'Real Time' with a speed of 20 Mega-pixels per second.

This report describes the research for application possibilities of the Optimized Pixel Template Image Correlator (OPTIC) chip. In chapter 2 a description of the OPTIC is given. The chapter discusses the main principle of operation. Chapter 3 describes a testcard which is build to perform this investigation and the control software for it. In chapter 4 the OPTIC application and some examples are described. The last chapter contains the conclusions of the assignment.

2 OPTIMIZED PIXEL TEMPLATE IMAGE CORRELATOR

2.1 OPTIC architecture

The OPTIC is able to execute a maximum/minimum sorting algorithm at a speed of 20 MHz. Every clock cycle, it can calculate the maximum, minimum or combinations of both from the internally stored pixels. The architecture is divided into four main blocks : Configurable Shift Register Array, Max-Min Sorting Array, Summation and Comparison Circuity and Controller (see figure 2.1). These four blocks will be explained in the next four paragraphs.

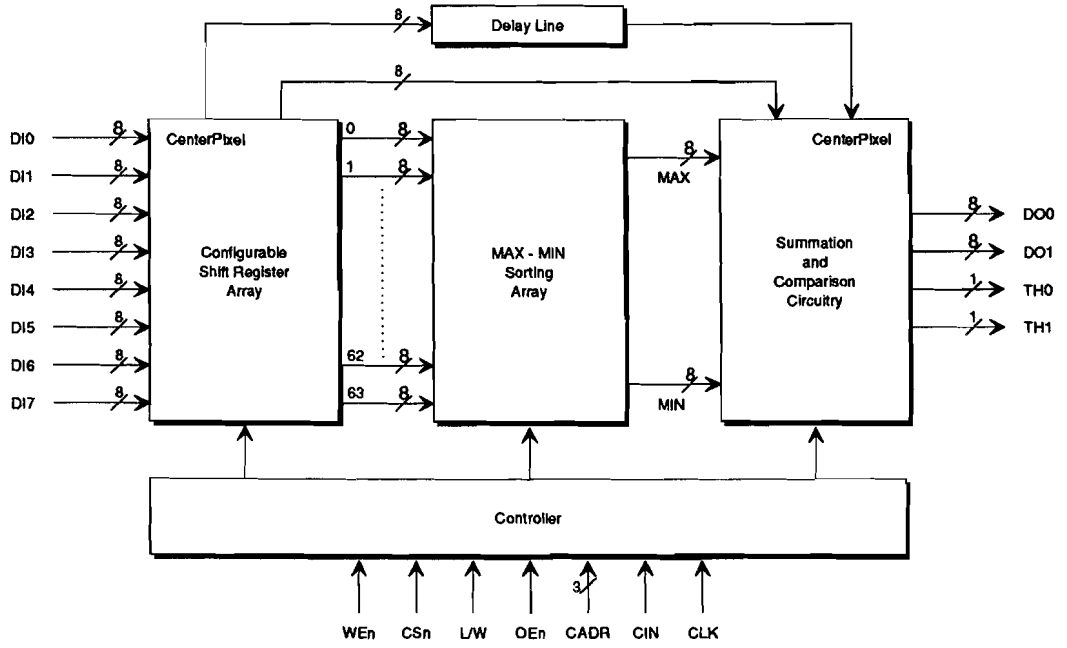


Figure 2.1 *Blockdiagram of the OPTIC*

2.2 Configurable Shift Register Array

The Shift Register Array, as shown in figure 2.2, contains sixty-four 8 bit registers. These registers are joined together in eight blocks of eight registers and form a Shift Array. Multiplexers at the beginning of each row allow data from a previous row or data from a Data Input (DI) to be routed

to the Shift Array. This allows a pixel processing window to be configurated in four different shapes, 1x64, 2x32, 4x16 or 8x8.

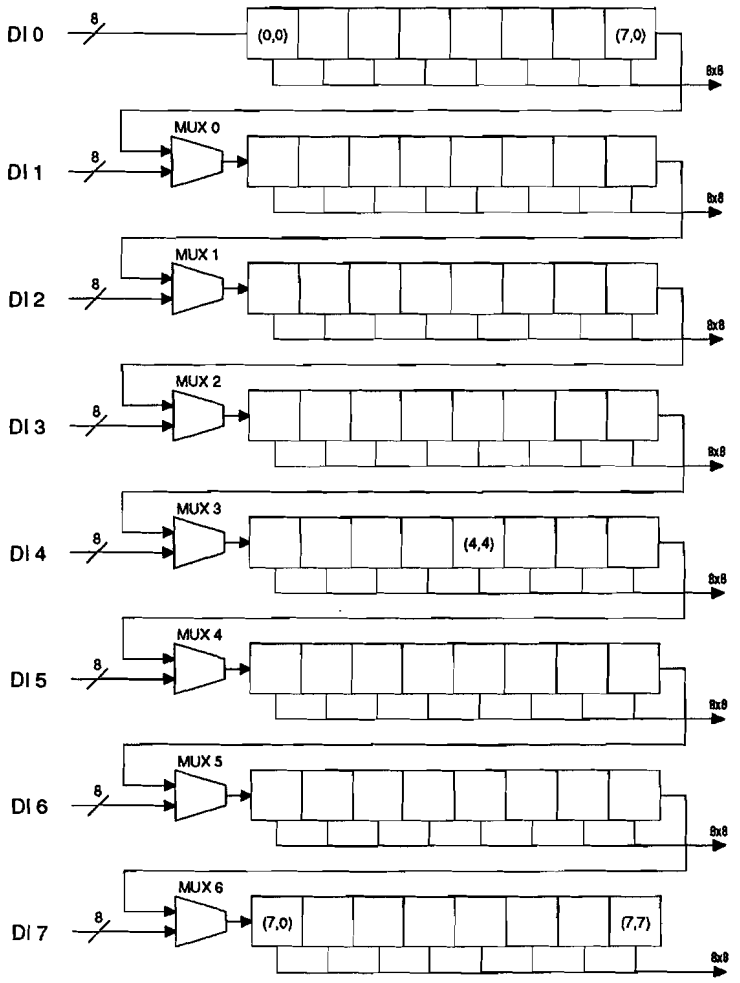


Figure 2.2 *Configurable Shift Register Array*

2.3 Max-Min Sorting Array

Each clock cycle the Sorting Array in figure 2.1 takes 64 pixels from the Shift Array. The maximum and minimum pixels are sorted out by a 6 pipeline deep arrangement of ninety-four cells (see figure 2.3). The basic cell required to perform this function consists of two inputs and two outputs. The inputs of each cell are two pixels. The maximum of these two is routed to one output and the minimum is routed to the other output.

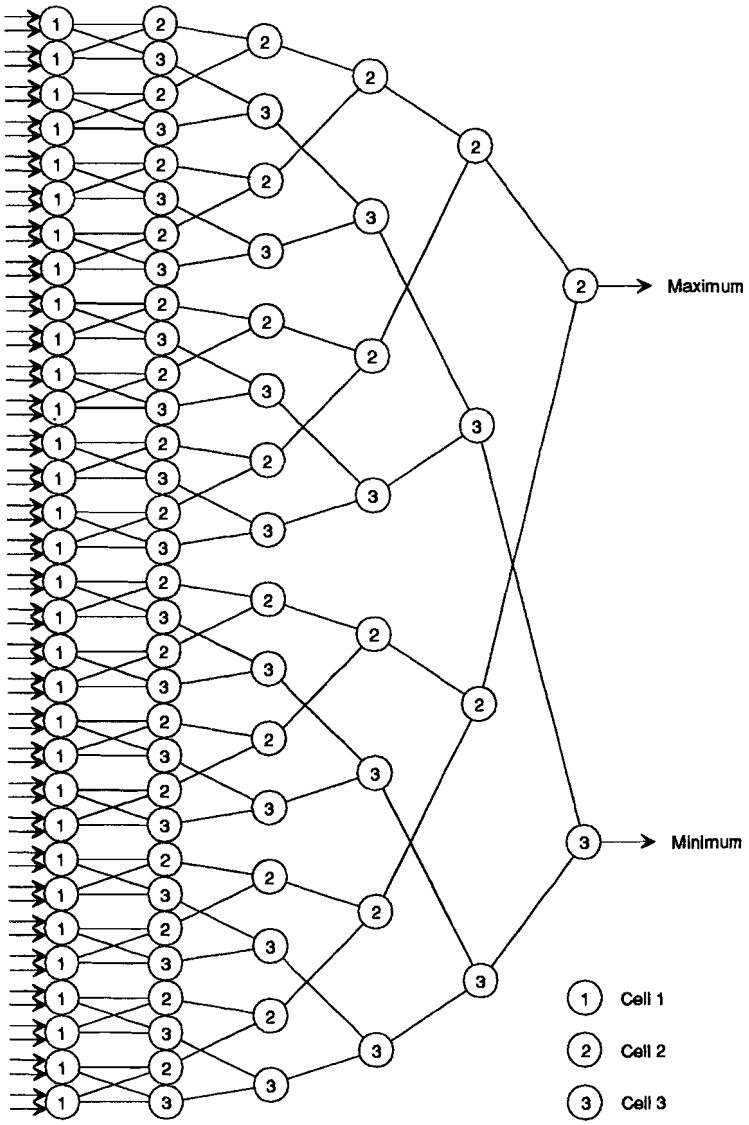


Figure 2.3 *Max-Min Sorting Array using different cell types*

However to make the OPTIC suitable for different applications, more sophisticated cells have been implemented. For some of the applications it will be necessary that some pixels inside the window can be excluded from the sorting process (don't cares). For other applications the possibility to sort the minimum or maximum of a certain group inside the window has been implemented. As a result of this each pixel can have four attributes : a pixel can belong to the maximum group (MAX) or the minimum group (MIN), being considered for both maximum and minimum sorting (FIL) or being excluded from both groups (DC). This results in a two bit user programmable code for each pixel.

Next to these pixel attributes it is even possible to load three other sets of attributes in the OPTIC. The right set can be selected by an external control address.

In figure 2.3, the first column of the Sorting Array consists of thirty-two 2 input sorters. A set of rules which depends on the pixel attribute, controls the operation of this first column of cells (further referred to as Cell 1). The twelve decision rules of these cells are shown in figure 2.4. The normal sorting is given in the last rule where two pixels of attribute FIL are input to a sorter. In this case the maximum is routed to the top output and the minimum to the bottom output. In figure 2.4 the maximum is always routed to the top and the minimum to the bottom.

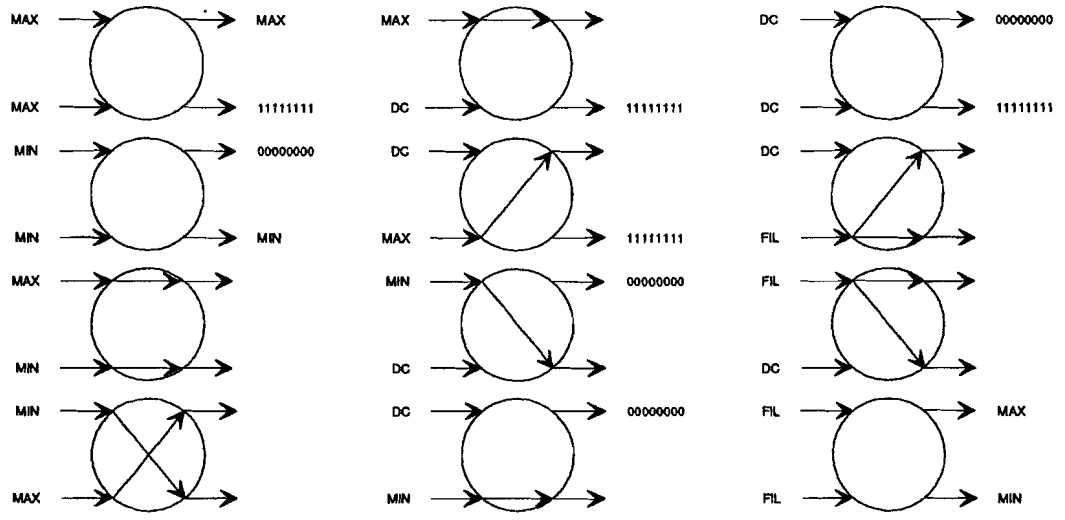


Figure 2.4 *Cell 1 decision rules*

In another example both inputs to the sorter are set to DC. In this case the 'maximum' output is set to zero and the 'minimum' output is set to 255 (maximum allowed greyvalue in an image). This ensures that these pixels do not influence the rest of the pixel sorting process.

As shown in figure 2.5 Cell 1 contains a normal Max-Min sorter but the multiplexers which are normally controlled by a comparator are now also controlled by a decoder. The register in front of the decoder contains the attribute information about the two connected inputs (4 bits). This attribute information is decoded and the cell is configurated according to the decision rules in figure 2.4. The corresponding attribute registers of each Cell 1 are combined in one 128 element shift register which can be externally controlled and loaded (see sub-paragraph 2.5.1). Depending on the address ATADDR[1..0] Cell 1 will use one of the four pixel attributes.

Once the pixels are sorted in these first cells, the remaining sorting is performed with two simpler cells. Cell 2 of figure 2.5 determines the maximum of the two input pixels and Cell 3 determines the minimum. The remaining of the Max-Min Sorting Array is built with these two cell types (see figure 2.3).

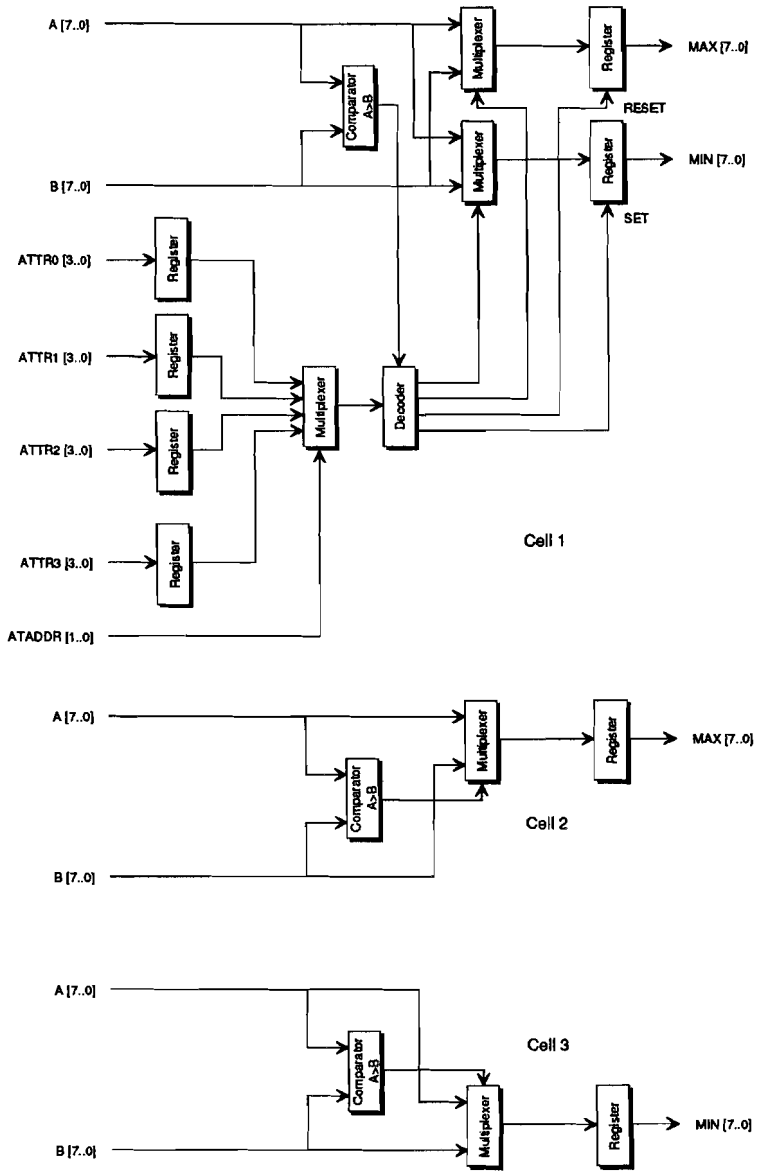


Figure 2.5 *Sorting cell architecture*

2.4 Summation and Comparison Circuitry

The Summation and Comparison Circuitry as can be seen in figure 2.1 combines the maximum and the minimum values sorted out by the Sorting Array, and routes them to the filter outputs according to a selected MODE. Table 2.1 lists the supported modes.

Table 2.1 *Supported operation modes*

MODE	C[15..13]	DO0	DO1
0	000	MIN	CPNF
1	001	(MAX + MIN) / 2	CPNF
2	010	MAX	CPNF
3	011	DIFF	CPNF
4	100	MIN	MIN
5	101	(MAX + MIN) / 2	(MAX + MIN) / 2
6	110	MAX	MIN
7	111	DIFF	(MAX + MIN) / 2

DO0 and DO1 are 8 bit greyvalue outputs, C[15..13] are bits of the configuration chain and DIFF and CPNF are defined as follows :

DIFF = (MIN - MAX) if C[4] = 0
DIFF = (MAX - MIN) if C[4] = 1
CPNF = Centre Pixel if C[16] = 0
CPNF = Noise Filtered output if C[16] = 1

These different modes are selectable in a user defined configuration chain which can be loaded in the OPTIC (see sub-paragraph 2.5.1).

The two other OPTIC outputs TH0 and TH1 (see figure 2.1) are determined as follows :

TH0 = DIFF ≥ Threshold
TH1 = (MAX=CP) && (CP<>0) if C[17] = 1

The Centre Pixel or CP used in table 2.1 and figure 2.1, is the pixel located at register (4,4) in the Shift Register Array (see figure 2.2). This CP is delayed the same number of clock cycles as the Sort Array (6 cycles) needs to sort out the maximum and minimum. The delay function is performed by the Delay Line in figure 2.1.

2.5 Controller

The Controller of figure 2.1 controls the operating modes LOAD and WORK. The modes are selectable by the L/W input of the OPTIC. In LOAD mode all control data can be downloaded into the filter, i.e. window shape, mode, pixel attribute and threshold data. During this mode the outputs of the filter are in High-Z state and Out Enable (OEn) is disabled. All control data will be fixed when the OPTIC is changed to WORK mode. During this mode Write Enable (WEn) is disabled.

During this WORK mode the outputs of the chip are controlled by OEn and the two LSB bits of CADR determine which set of pixel attribute data will be used. There is also a chip select (CSn) available for selecting the chip.

2.5.1 Loading of the template and configuration data

If the OPTIC is in LOAD mode all control data can be loaded into the processor. The data is loaded serially through the CIN input from the OPTIC. This process is controlled by the WEn signal and is done synchronous to the clock (CLK). There are five chains to be loaded, four template/template chains (nr. 1..4) and one configuration chain (nr. 5). The construction of these chains is shown in figure 2.6.

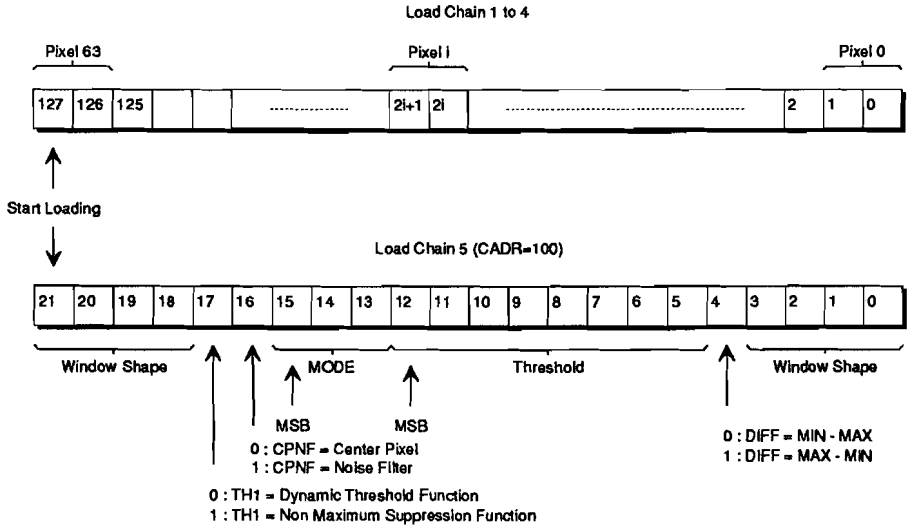


Figure 2.6 *Construction of load chains for the OPTIC*

The control bus, CARD, determines which data chain will be loaded. The different chains can be addressed using CADR as listed in table 2.2.

Table 2.2 *OPTIC load chain addressing*

CADR	Control Chain
000	Template 0
001	Template 1
010	Template 2
011	Template 3
100	Configuration

Each of the template chains is 128 bits long : there are 64 pixels per template and each pixel requires two bits to describe which of the four possible attributes are attached to that pixel. As a result of this it needs 128 clock cycles to load one template chain in the OPTIC. Table 2.3 describes the possible pixel attributes and gives the bit configuration for selecting the attributes.

Table 2.3 *Possible pixel attributes*

Attribute	MSB	LSB
MAX	0	0
MIN	0	1
DC	1	0
FIL	1	1

The attribute data of pixel number 63 (i.e. the pixel corresponding to the first template pixel loaded in the OPTIC in a 8x8 template configuration) will be loaded first when loading a template chain, since this is the last pixel in the chain. Also the MSB bit of every 2-bit pixel attribute should be loaded before the LSB (see figure 2.7).

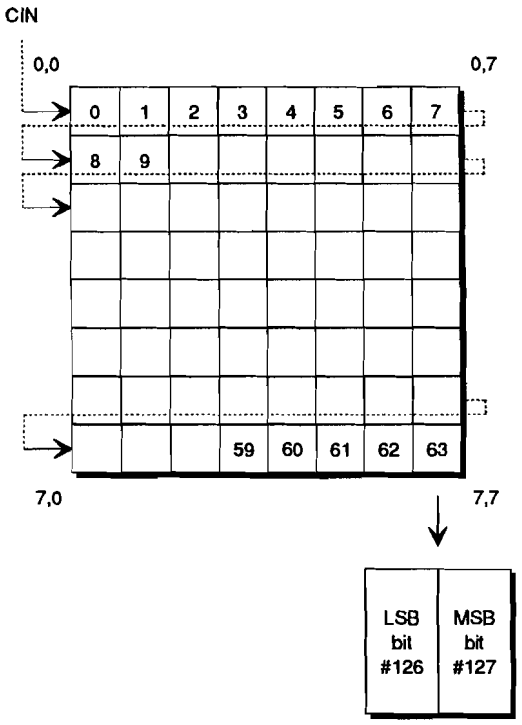


Figure 2.7 *Load chain mapped in window*

The configuration chain as shown in figure 2.6 is 22 bits long (c[21..0]) and contains the window shape, threshold and output select data. Bit c[0] is the first bit and is consequently loaded last. The first four and last four bits contain the window shape (see table 2.4).

Table 2.4 *Window configurations*

Window Shape	c[21..18]	c[3..0]	Window inputs
8x8	1111	1111	DI7 → DI0
4x16	0101	0101	DI6,DI4,DI2,DI0
2x32	0001	0001	DI4,DI0
1x64	0000	0000	DI0

The 8-bit positive threshold is located in bits c[12..5]. The remaining bits in this configuration chain determines the OPTIC output and allows implementation of various algorithms. The definition of bits c[15..13] is already described in table 2.1.

3 HARDWARE DESCRIPTION

3.1 PAPS system

To test the OPTIC and to evaluate its applications, a test board is developed. This board fits into the Picture Acquisition and Processing System (PAPS) from Philips. This system is a inhouse development for high-end machine vision applications. Figure 3.1 shows the PAPS system configuration which is used for testing the OPTIC applications.

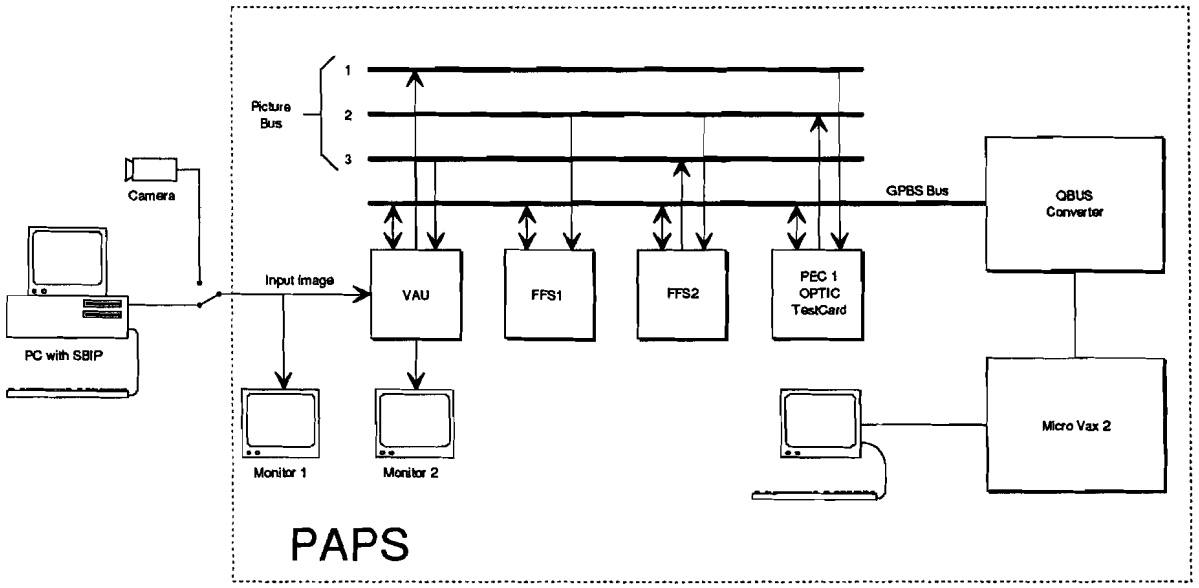


Figure 3.1 *The PAPS system setup*

The input pictures for the PAPS system can be generated by the PC (artificial or stored on disk) or by a CCD camera (live images). Monitor 1 shows the input image.

The PAPS system is configurated with the following hardware cards :

VAU : Video Acquisition Unit that samples the incoming image and generates the appropriate timing signals on the control channel of the Picture Bus.

FFS1, FFS2 : Full Frame Store to store intermediate images. These images are the results of other operations.

PEC 1 : PAPS Experimental Card. This board contains two OPTIC chips and some logic for controlling the chip.

The Picture bus (PI-bus) transports images between the different cards. The incoming image is put on PI-bus 1 by the VAU. The PEC-card with the OPTICs uses this image, processes it and places the result on PI-bus 2. The result can than be stored in FFS2 which places the frozen image on PI-bus 3. The VAU finally displays the image (the result of the performed OPTIC operation) from PI-bus 3 on monitor 2.

3.2 OPTIC testcard

The OPTIC testcard contains all the hardware developed to test the OPTIC applications. Figure 3.2 shows the block diagram of the testcard.

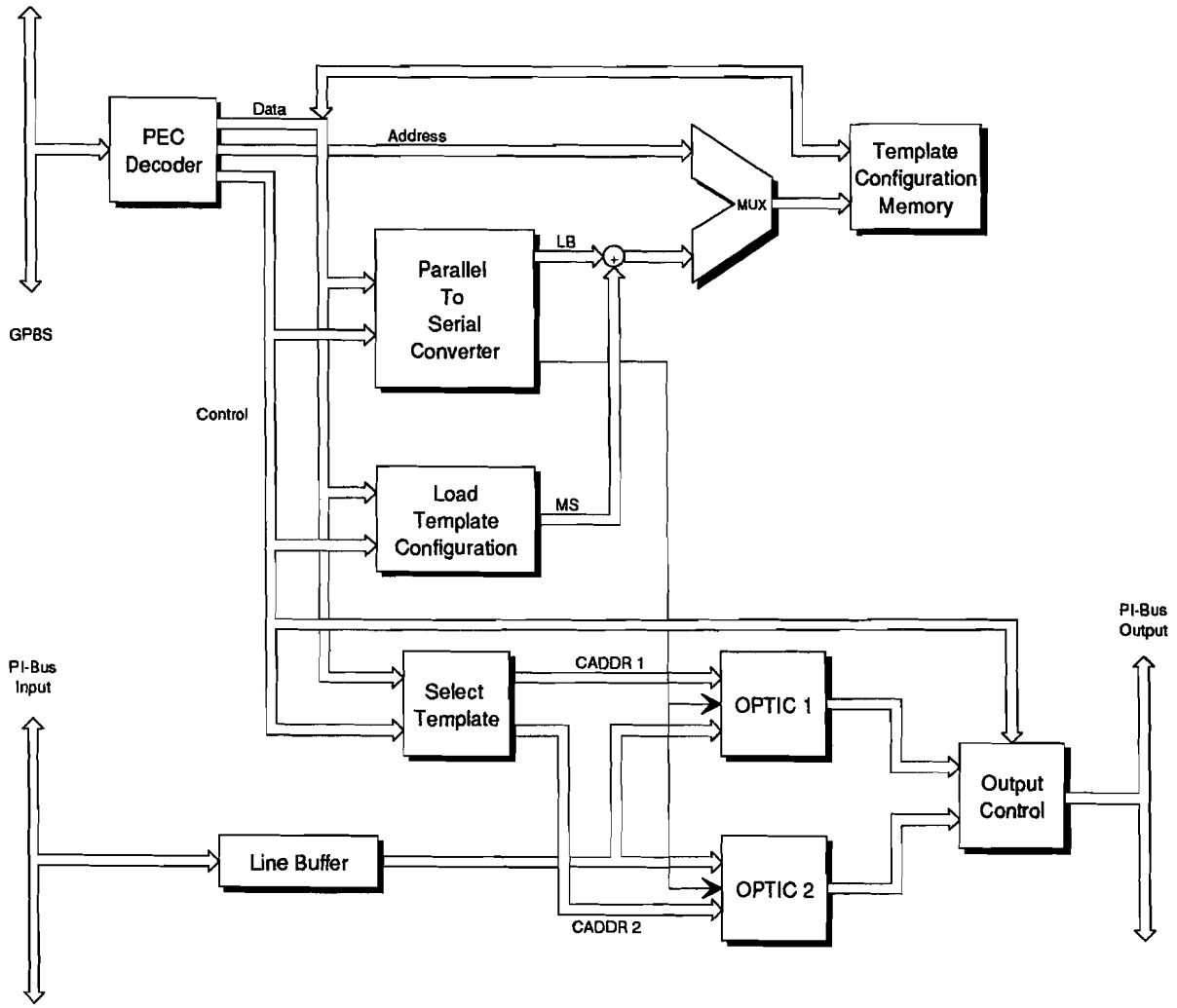


Figure 3.2 *Blockdiagram of the OPTIC testcard*

The next sub-paragraphs describe the PEC Decoder, the Template Configuration Memory, the Parallel to Serial Converter, the Line Buffer and the Output Controller. The two other blocks are registers in which data can be placed. They control some processes on the card. This will be described in paragraph 3.3

3.2.1 PEC Decoder

The PEC Decoder is an interface between the PAPS system and the hardware on the card. It converts the signals from the GPBS bus to simple read and write pulses. With these signals it is possible to communicate with the registers on the PEC card. The decoder can also select the PI-bus from which the input image will be used and where the result will be displayed.

As shown in figure 3.2 the decoder controls the Address bus (16 bit), the Data bus (8 bit) and the Control bus. The control bus contains 7 write lines for clocking data from the Data bus in the different registers. This can be done on the host with the following instruction :

```
pwer (Caddr, Reg_nr, Data)          Write Data in Register Reg_nr
```

Caddr is the card-address and is selectable with the dip-switch MOD ADD on the PEC card. This address is user selectable but has to be different from the other card addresses in the PAPS system.

The selection of the output PI-bus can be done with the following instruction :

```
wer (Comaddr, Data)                 Select output bus
```

Comaddr is the common address of the PAPS system and is selectable with the dip-switch COMM ADDR on the PEC card. It has to match with the other common addresses of the other cards in the system. The Data used in the instruction consists of two parts as listed in table 3.1

Table 3.1 *Data configuration*

D[15..8]	D[7..0]	Description
0 0 0 0 1 0 0	x x x x x x x	Output PEC to PI-bus 1
0 0 0 1 0 0 0	x x x x x x x	Output to PI-bus 2
0 0 1 0 0 0 0	x x x x x x x	Output to PI-bus 3
x x x x x x x	Caddr	Card address PEC-card

The Address bus in figure 3.2 is only used for writing data in the Memory. This can be done with the following instructions :

```
p800alloc(0,1,'B');                    Allocate memory space
pwer(Caddr, 0, \H20);                  Enable writing to Memory

mwbyte (Address, Data 1);
mwbyte (Address, Data 2);
         :                             Write data in Memory
mwbyte (Address, Data n-1);
mwbyte (Address, Data n);

pwer(Caddr, 0, 0);                     Disable writing to Memory
```

The first instruction (p800alloc) assigns some virtual address space on the host to the memory on the card. The first pwer instruction enables the writing of the memory on the card. The last one disables the writing to the memory.

3.2.2 Template Configuration Memory

The size of the Memory is 2Kbx8 bytes and it is organized in 128 blocks of 16 lines. Figure 3.2 shows the organisation of a part of the memory. Every block contains configuration data or template data.

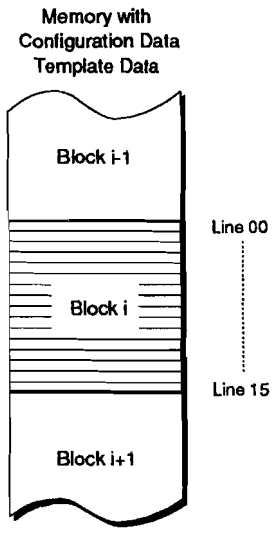


Figure 3.3 *Memory organisation*

Blocks 0..15 contain 16 configuration chains and blocks 16..127 contain the pixel attributes of 112 templates. This configuration and template data will be written in the memory with the instructions mentioned in paragraph 3.2.2. During this writing process the memory address is generated by the decoder. If the OPTIC is loaded with one of the blocks from the memory the MS Address will be

generated by the Load Template Configuration block in figure 3.2. This register (REG 6) can be filled with the block address. The LS part of the address will be generated by the Parallel To Serial Converter. The output data of the memory containing the template and configuration will also go to this part of the testcard. Figure 3.4 shows how the pixel attributes of a template and the configuration data are mapped in a memory block.

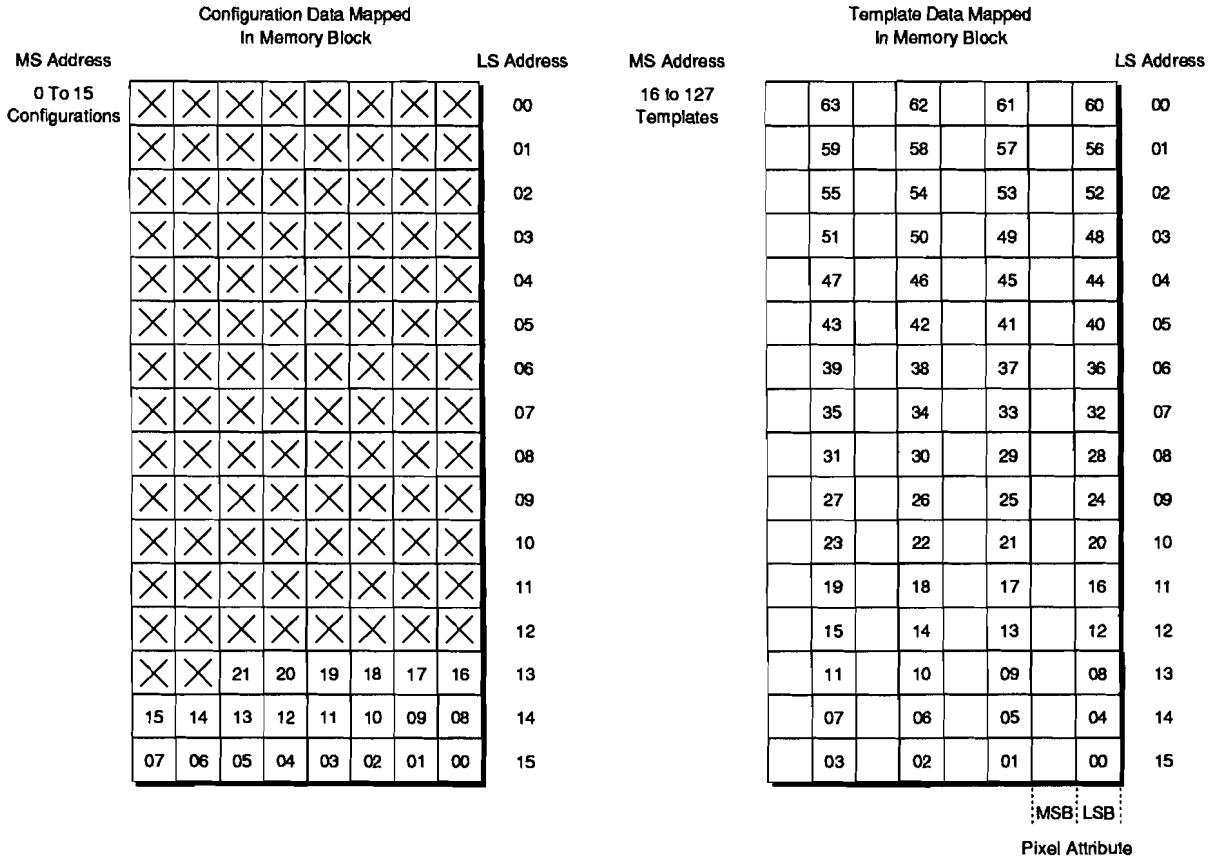


Figure 3.4 *Data mapped in memory blocks*

3.2.3 Parallel To Serial Converter

The Parallel To Serial Converter converts the parallel data from the memory in a serial bitstream. This bitstream will be loaded into the OPTIC. Appendix A shows the circuitry of the Converter and in figure 3.5 the timing diagram can be seen.

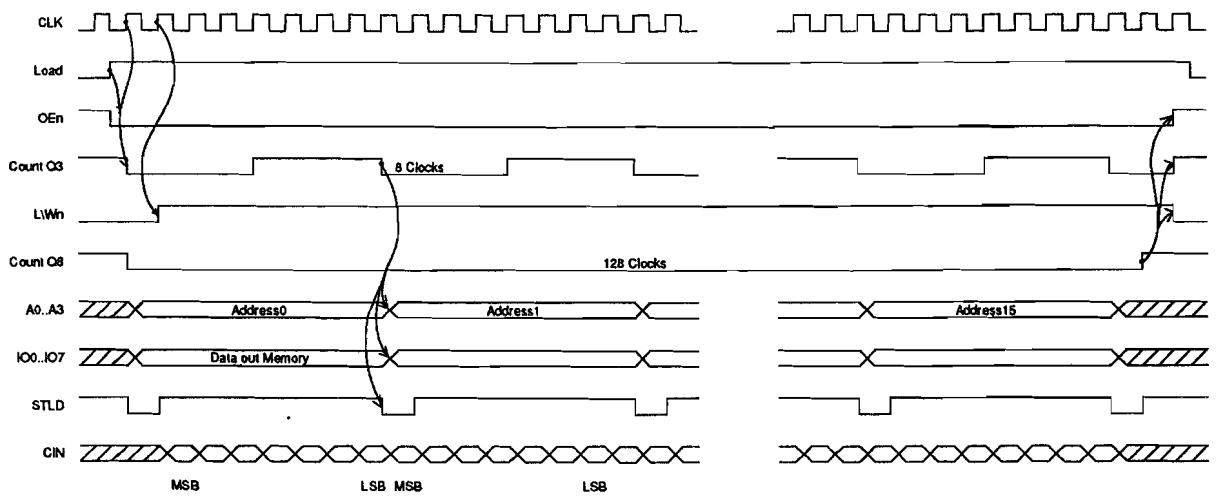


Figure 3.5 *Timing diagram of the Parallel to Serial Converter*

The conversion process is started with LOAD going high and OEn going low. On the rising edge of the next clock a counter is enabled and the OPTIC is changed from WORK mode to LOAD mode (L\W going high).

Q[7..4] of the 16 bits counter generates the LS address for the memory (A[3..0] : addresses one of the 16 lines of a block). The data (IO[7..0]) is clocked in the shift register at the falling edge of Q8 (STLD going low). During the next 8 clock cycles the data is shifted out of the register (MSB first and LSB last). After 8 cycles a new address is generated and the next line will be shifted out of the converter.

After 128 clock cycles (Q8 going high) one complete memory block (16x8 bits) is shifted out the Converter. As a result the OPTIC will change back to WORK mode (LW going low) and the LOAD and the OE signal can go high.

3.2.4 Line Buffer

As described in paragraph 2.2 the OPTIC can use different shapes of pixel processing windows. While processing an image the window moves along the screen as shown in figure 3.6. The result of the process is placed on position (7,7) in the screen. The resulting image will therefore be shifted in both x and y directions. The size of the shifting depends on the position of the used pixels inside the window (see figure 3.7). As can be seen in figure 3.7 the shifting will be minimized if the pixels used are around position (7,7) in the window.

Next to the above mentioned delay, there is also the latency through the chip. The amount of latency is determined by the shape of the used window. The number of clock cycles delay (pixels delay) is given by the following equation :

$$\textit{Latency} = (\textit{Window\_delay}) + 9)\ \textit{clock cycles} \qquad (3,1)$$

In the above equation the Window_delay is 8 cycles for the 8x8 window, 16 for 4x16, etc... ;

During this application research only a window of 8x8 pixels is used. This means that the OPTIC needs, every clock cycle, 8 new pixels from 8 different lines.

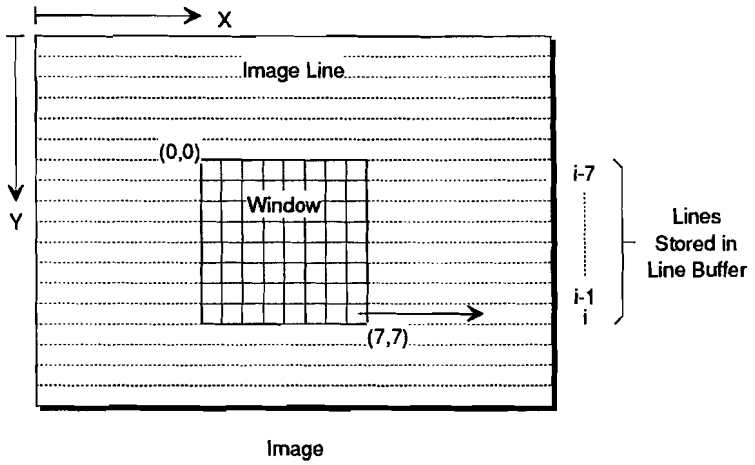


Figure 3.6 *Main principle of pixel processing.*

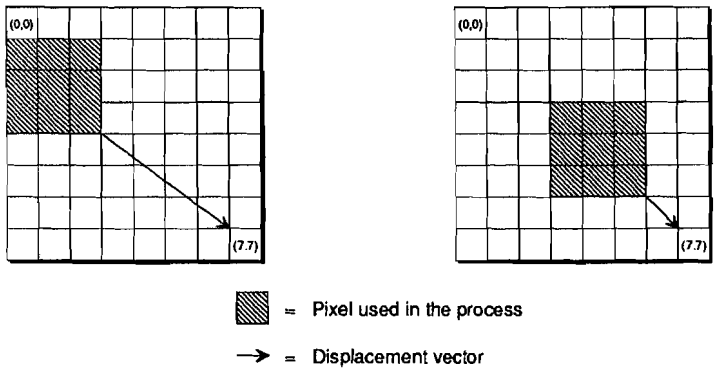


Figure 3.7 *Shifting of the processed result opposite to the used pixels of a window*

The Line Buffer as shown in figure 3.2 stores 7 image lines in its memory and generates 7 delayed pixels belonging to the 7 lines. To read and write the Line Buffer, the method of Read Modify Write is used. Figure 3.8 shows the blockdiagram of a part of the Line Buffer. The Buffer used has 8 memories and 8 registers. Figure 3.9 shows the timing belonging to the circuitry.

For the RMW-cycle 3 clocks are needed. Figure 3.9 shows these clock signals. The only difference between the signals is a fixed phase shift. This shift is determined by the setup time of the address counter and the propagation delay of the used registers. The timing circuitry is drawn in appendix A.

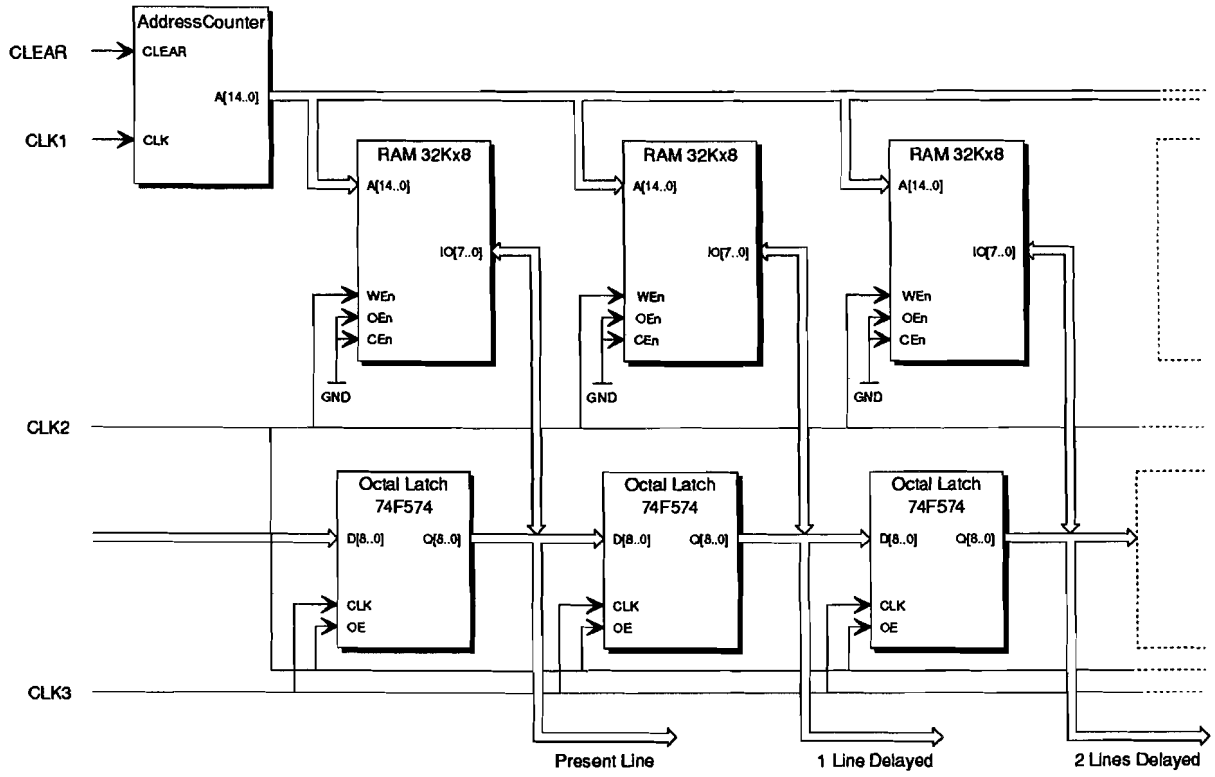


Figure 3.8 *Blockdiagram of the Line Buffer*

Both memory cycles (read and write) are WE (Write Enable) controlled. This means that when WE is low the memory can be written and when WE is high the memory can be read. CLK1 is used to generate the address, CLK2 is used to enable the memories and the Latches (Output Enable) and CLK3 is used to latch new data in the registers. Since the WE and OE are level signals the following will happen :

- On the rising edge of CLK1 the new address is generates, the outputs of the registers are disabled.

- Before CLK1 goes low, CLK3 goes high and the data on the data bus and the output of the memories is latched in the registers.
- CLK2 goes low, so the outputs of the registers are enabled and the memories can be written.

3.2.5 Output Controller

The outputs of the two OPTICs are the input of the Output Controller in figure 3.2. The Controller selects which output (DO0, DO1, TH0, TH1) will be displayed on the PI-bus of the PAPS system. It is also possible to compare the output of one OPTIC with a user programmable threshold. This result will in a binary output image.

The Output Controller which is programmed in a EPLD is shown in appendix A. The Controller is programmable by placing control data in register 5. In the next paragraph all the control registers are described.

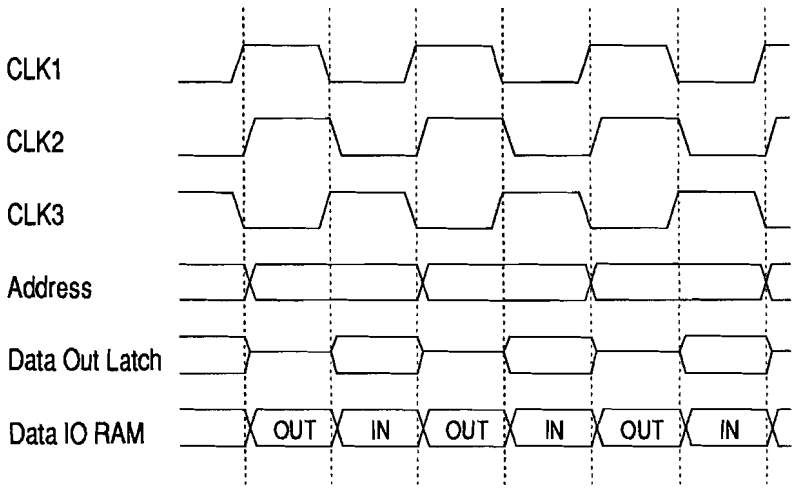


Figure 3.9 *Timing Read/Write cycle of the Line Buffer*

3.3 Control registers OPTIC testcard

The testcard contains 4 user programmable registers (REG0, REG5, REG6, REG7) which can be written from the host with the instruction mentioned in paragraph 3.2.1. With the data in these registers it is possible to control the functions of the testcard. The function of these registers is listed in table 3.2. A detailed description of the Control registers is given in appendix B.

Table 3.2 *Function of the Control registers*

Register	Function
REG 0	Controls the PEC-decoder
REG 5	Controls the Output Controller
REG 6	Controls the serial load process
REG 7	Controls the two OPTICs

REG 0

With the bits in this register it is possible to change the input of the card. Setting one bit of the register starts the handshake protocol between the card and the host.

REG 5

With the data in this register the outcome of the Output Controller can be influenced. It is possible to select which output from which OPTIC will be displayed on the PI-bus. The data also contains the threshold for making a binary output image.

REG 6

The data in register 6 controls the loading of data in the two OPTICs. Setting one bit starts the parallel to serial conversion and the loading process of one data block in the Memory. The register also contains the block address of the block that will be loaded in the OPTIC.

REG 7

Setting one bit in this register changes the OPTIC from WORK mode to LOAD mode. The register also contains the CADDR. This address selects in LOAD mode the type of chain that will be loaded. In WORK mode CADDR selects one of the four processing windows.

4 OPTIC APPLICATIONS

4.1 Introduction to Max-Min filtering

The OPTIC uses two pixel processing techniques. It can calculate the maximum and minimum of a group of pixels. These two operations are the basis of all the applications. The OPTIC uses the pixels inside a window (W_{fil}) and processes them. The window moves along the image as described in paragraph 3.2.4. The result of the sorting operation is placed at position (x,y) (see figure 4.1).

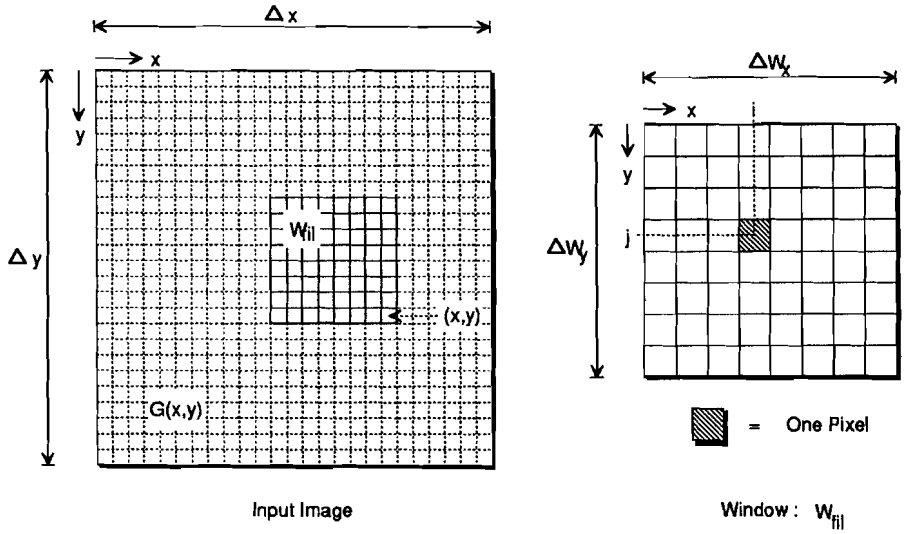


Figure 4.1 *Image and window variables*

The window contains a template. This template determines which pixels will influence the outcome of the filter operation. There are two kind of filters : a maximum filter (T_{max}) and a minimum filter (T_{min}). These are the basic filters used in all the applications. Equation (4,1) contains the definition of the values of the pixels inside the two windows.

$$T_{max}(x,y) = \begin{cases} \textbf{max} & \rightarrow \textit{do care pixel} \\ \textit{dc} & \rightarrow \textit{don't care pixel} \end{cases}$$

$$T_{min}(x,y) = \begin{cases} \textbf{min} & \rightarrow \textit{do care pixel} \\ \textit{dc} & \rightarrow \textit{don't care pixel} \end{cases} \qquad \textbf{(4,1)}$$

Equation (4,2) defines the result of a template operating on an image in position (x,y). This operation will be written as $G(x,y) \odot W_{fil}(i,j)$ with W_{fil} either T_{max} or T_{min}.

$$G(x,y) \odot T_{max}(i,j) = \begin{cases} G(x,y) & \text{if } T_{max}(i,j) = \text{max} \\ 0 & \text{if } T_{max}(i,j) = dc \end{cases}$$

$$G(x,y) \odot T_{min}(i,j) = \begin{cases} G(x,y) & \text{if } T_{min}(i,j) = \text{min} \\ 255 & \text{if } T_{min}(i,j) = dc \end{cases} \qquad (4,2)$$

The result of the above definition is that only the pixels of the image which match with the do care pixels in a certain template are used for the calculation of the output pixel at position (x,y).

4.2 Morphological filters

Morphological filters provide an approach to the processing of digital images. There are two kind of morphological filters: erosion and dilation. These filters remove noise from an image. If the two basic filters are combined in a cascade structure more complex filters can be build. The filters can be made for a certain application. The operations performed by these complex filters are called opening and closing of an image.

The next few sub-paragraphs the erosion and dilation filters will be described. In paragraph 4.3 the more complex filters will be discussed.

4.2.1 Erosion

The morphological operation erosion removes pixels from objects in an image. The shape of the object roughly remains the same. The operation uses the pixels of G(x,y) which lie inside the template T_{max} and calculates the maximum of these pixels. These maximum pixels form the eroded output picture. Equation (4,3) defines the mathematical operation $G \ominus T_{max}(x,y)$ at position (x,y) of the image.

$$G \ominus T_{max}(x,y) = \underset{i=0}{\overset{\Delta W_x}{MAX}} \; \underset{j=0}{\overset{\Delta W_y}{MAX}} G(x-i,y-j) \odot T_{max}(\Delta W_x - i, \Delta W_y - j) \qquad (4,3)$$

This erosion operation is performed at every position (x,y) in the image. The eroded output image $G_{erosion} = G \ominus T_{max}$ can thus be written as a set of pixels $G \ominus T_{max}(x,y)$ (see equation (4,4)).

$$G \ominus T_{max} = \{G \ominus T_{max}(x,y) \mid x \in [0,\Delta x] \wedge y \in [0,\Delta y]\} \qquad (4,4)$$

Figure 4.2 gives an example of erosion on a white image with black objects. The result of the operation is, that the dark areas become smaller (1 pixel in every direction) and the white areas grow larger.

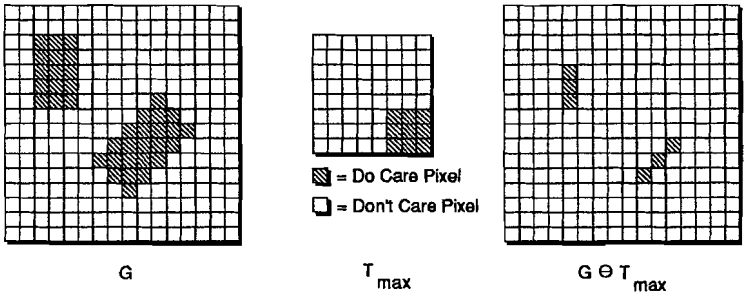


Figure 4.2 *Example of erosion on an image*

4.2.2 Dilation

The morphological operation dilation adds pixels to the objects in an image and is therefore the opposite operation of erosion. The operation also uses the pixels of the image which match with the do care pixels of the template T_{min}. Only now the minimum of those pixels is calculated. The result of this operation (see figure 4.3) is that the dark areas grow larger and the white areas become smaller.

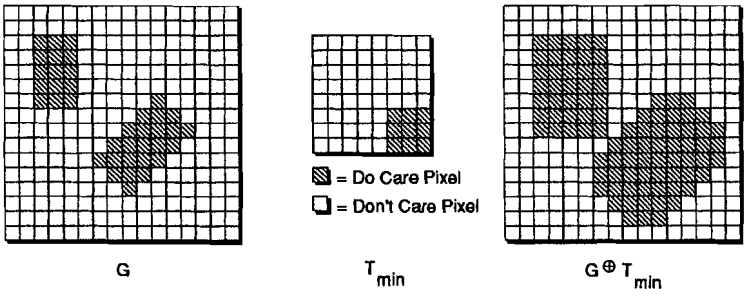


Figure 4.3 *Example of dilation on an image*

Equation (4,5) shows how the minimum of the pixels at position (x,y) can be calculated.

$$G \oplus T_{min}(x,y) = \underset{i=0}{\overset{\Delta W_x}{MIN}} \; \underset{j=0}{\overset{\Delta W_y}{MIN}} \; G(x-i,y-j) \odot T_{min}(\Delta W_x - i, \Delta W_y - j) \qquad (4,5)$$

The dilation operation is performed at every position (x,y) in the image. The output image $G_{dilation} = G \oplus T_{min}$ can thus be written as the set of pixels $G \oplus T_{min}(x,y)$ (see equation (4,6)).

$$G \oplus T_{min} = \{G \oplus T_{min}(x,y) \mid x \in [0,\Delta x] \wedge y \in [0,\Delta y]\} \qquad (4,6)$$

If dilation or erosion is performed on white objects with a dark background, the definitions of equation (4,3) and (4,5) will be exchanged. Erosion of a black object will become the same as dilation of a white object (the background in figure 4.2 grows larger) and dilation of a black object will become the same as erosion of a white object (see the background in figure 4.3).

The two morphological operations can be performed with the OPTIC using the configuration and template listed in table 4.1. With this window an erosion and dilation operation is performed on the image in figure 4.4.

Table 4.1 *Example of the implementation of erosion and dilation*

Template pixel attributes								Configuration	Description
x	x	x	x	x	x	x	x	Window shape	8x8
x	x	x	x	x	x	x	x	NMS	Don't care
x	x	x	x	x	x	x	x	CPNF	Don't care
x	x	x	x	x	x	x	x	MODE	DO0 = Max
x	x	x	x	x	x	x	x		DO1 = Min
x	x	x	x	x	**f**	**f**	**f**	Threshold	Don't care
x	x	x	x	x	**f**	**f**	**f**	DIFF	Don't care
x	x	x	x	x	**f**	**f**	**f**	TH0	Don't care
								TH1	Don't care

In the image the operations are performed on white objects. The two templates T_{min} and T_{max} used to perform the two operations are the same.

When the Minimum output from the OPTIC (DO0) is displayed on the monitor the eroded image can be seen. The Maximum output (DO1) shows the dilated image. The two output images are given in figure 4.5 and figure 4.6. It is easy to see that there is one pixel eroded (erosion) and one pixel added (dilation) in each direction. In general the amount of pixels eroded or dilated is equal to the amount of pixels between the CenterPixel of the do care pixels in the template and the boundary of the do care pixels. In this filter the distance is 1 pixels and therefor one pixel is eroded.

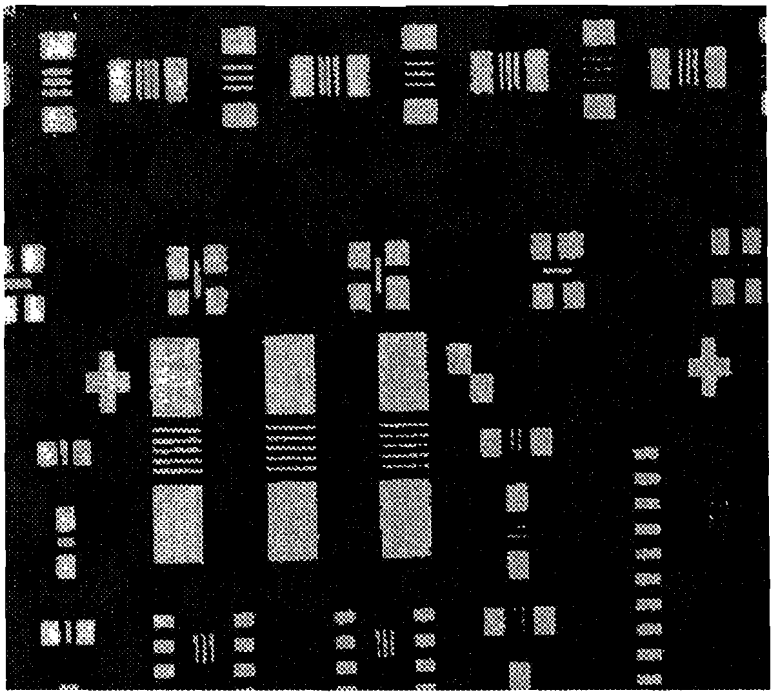


Figure 4.4 *Original image before filtering*

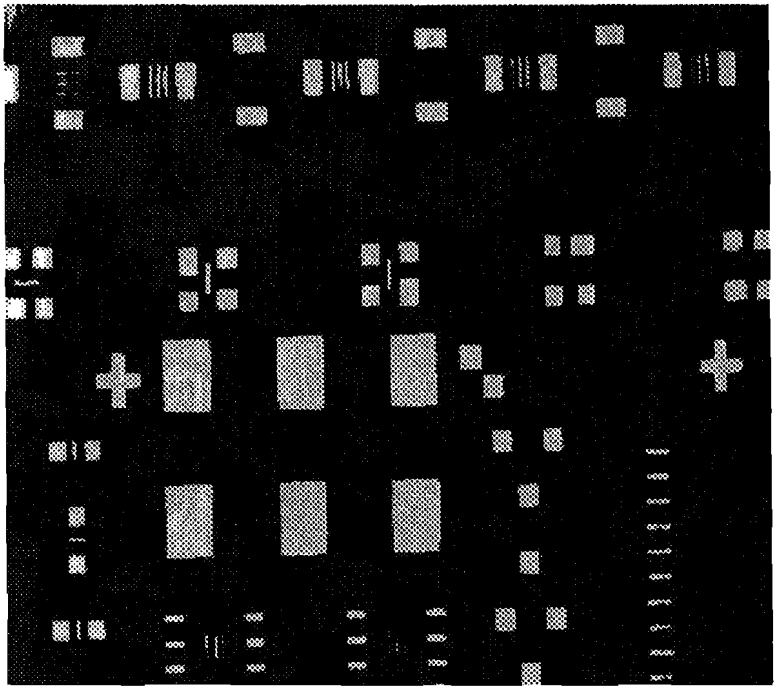


Figure 4.5 *Resulting image after erosion of the white objects*

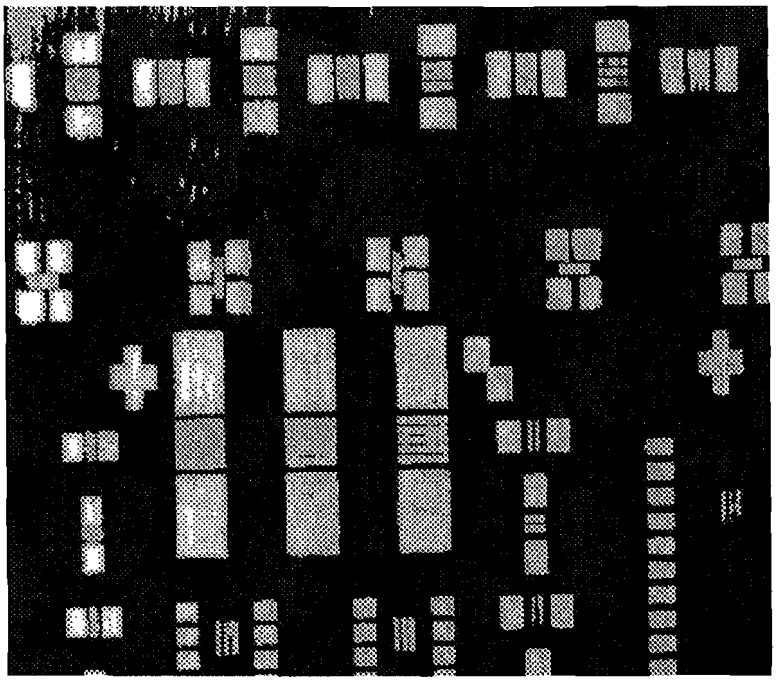


Figure 4.6 *Resulting image after dilation of the white objects*

The fact that both the morphological operations use a filter that calculates the minimum or maximum make the filters more or less independent for the amount of light received by the camera and of the focusing of the camera. Changing the amount of light or defocusing the camera doesn't influence the result. The local maximum or minimum remains at the same position and only the size changes. The result of defocusing the camera will be that the image becomes darker.

4.2.3 Influence of the filter shape on the operation

As can be seen in figure 4.3 the shape of the dilated figure is not exactly the same as in the original figure. This is the result of the filter shape used. Examination of the responses of various shapes to triangular noise on horizontal, vertical and diagonal object boundaries shows that triangular noise on a horizontal or vertical object boundary is not removed by a 4-neighbourhood filter shape (see figure 4.7). Likewise, triangular noise on a diagonal object boundary is not removed by a 8-neighbourhood filter shape (see figure 4.8).

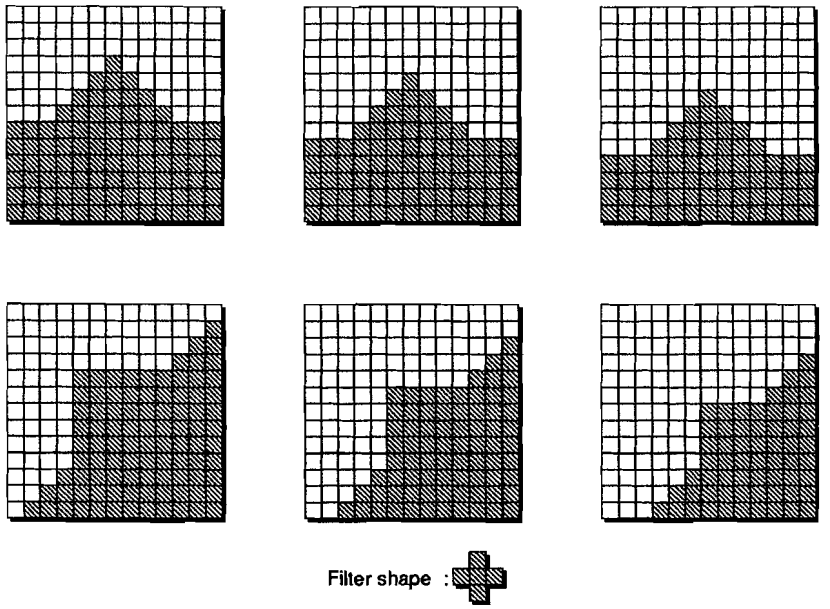


Figure 4.7 *Example of erosion with a 4-neighbourhood filter*

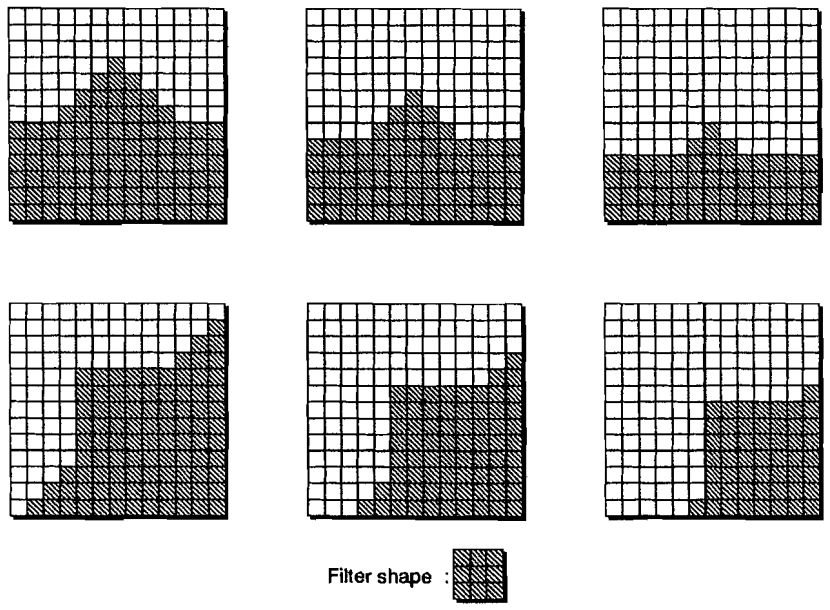


Figure 4.8 *Example of erosion with a 8-neighbourhood filter*

The 3x3 filter is therefor only useful for an erosion or a dilation of objects with orthogonal boundaries. If more complex boundaries have to be filtered a circular filter shape would be ideal. This shape combines the advantages of both the 4-neighbourhood and 8-neighbourhood filter shape.
It removes or adds the same amount of pixels in every direction. If it is necessary to erode or dilate more pixels in an image, the ideal circular filter can be approached with the use of different smaller filter shapes (for example with a X-shape followed by a cross shape). This last technique is explained and proven in the next sub-paragraph.

4.2.4 Complex filter shapes

Besides the erosion or dilation of one pixel (3x3 matrix of do care pixels) it is also possible to use more complex filter shapes. This can be done in two ways. One way is to pass the image through two different filters and the second way uses a different filter shape. This principle can be expressed for erosion with the chain rule for erosion (see equation (4,7)).

$$\begin{aligned}(A \ominus B) \ominus C &= A \ominus (B \oplus C) \\ &= A \ominus D \quad \wedge \quad D = B \oplus C\end{aligned} \tag{4,7}$$

In the above equation A is an input image and B and C are two different templates. Template D is the result of the dilation of B with template C.

Next to the rule for erosion there is also a chain rule for dilation. This one is described in equation (4,8)

$$\begin{aligned}(A \oplus B) \oplus C &= A \oplus (B \oplus C) \\ &= A \oplus D \quad \wedge \quad D = B \oplus C\end{aligned} \tag{4,8}$$

In this equation A,B,C and D are all templates (see equation (4,7)).

The above mentioned relations are all proved by Haralick, Sternberg and Zhuang [4]. They described mathematical morphology for both binary and greylevel images.

With the chain rules it is now possible to see that an erosion of two pixels in every direction can be achieved with two windows of 3x3 do care pixels or with one window with 5x5 do care pixels (see figure 4.9). It is also possible to compose more complex filter shapes out of several simpler filters.

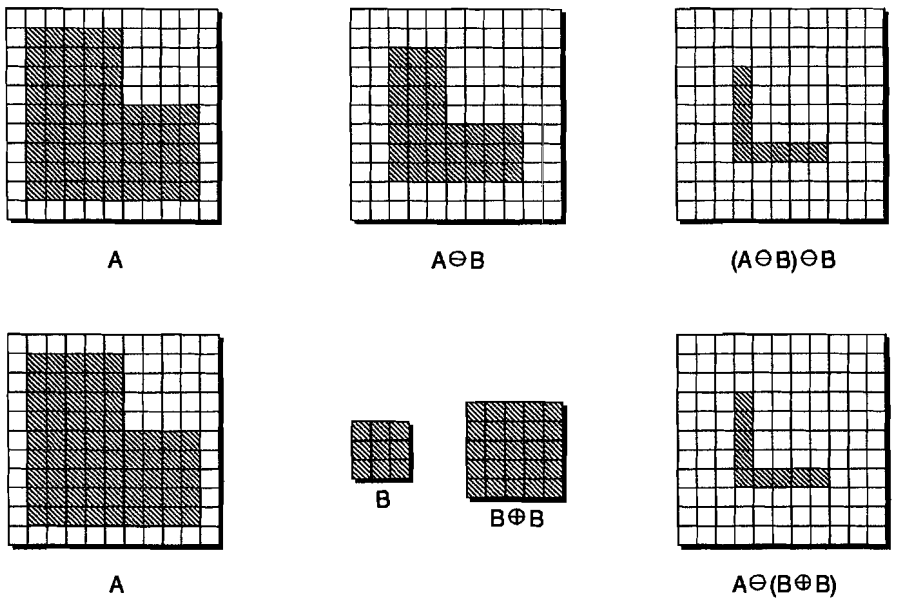


Figure 4.9 *Example of the chain rule for erosion of black objects*

A real example of the chain rule for erosion can be seen in figure 4.10. This figure shows the result of an erosion operation performed on the image from figure 4.4. The configuration of the OPTIC is the same as in table 4.1. However to erode more pixels a different template is used. The filter shape is a 5x5 matrix of do care pixels as listed in table 4.2. In this filter the amount of pixels between the CenterPixel of the matrix and the boundary of the filter is two pixels. This means that a 5x5 matrix removes 2 pixels from the objects in the image.

Table 4.2 *Pixel attributes of a 5x5 filter*

Template pixel attributes							
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	**f**	**f**	**f**	**f**	**f**
x	x	x	**f**	**f**	**f**	**f**	**f**
x	x	x	**f**	**f**	**f**	**f**	**f**
x	x	x	**f**	**f**	**f**	**f**	**f**
x	x	x	**f**	**f**	**f**	**f**	**f**

In comparing figure 4.10 with figure 4.5 it is easy to see that in the last figure more pixels have been eroded. The small vertical lines in figure 4.5 are now completely removed by the erosion operation.

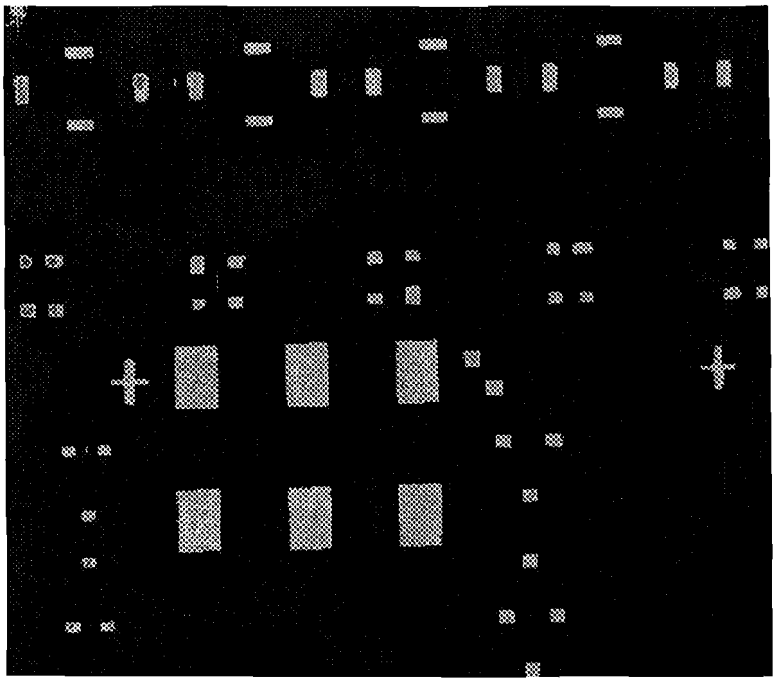


Figure 4.10 *Resulting output image after an erosion with a window of 25 do care pixels*

4.3 Noise filters

Noise filters are used to eliminate image details smaller than the filter-size without distortion of object boundaries. The noise filters are composed out of the erosion and dilation filters. The two filters are used in cascade. With this configurations two different ways of noise filtering are possible: opening and closing. Next to these last two filters there is also an edge-preserving noise filter. The three filters mentioned above will be described in the next sub-paragraphs.

4.3.1 Opening

Opening an image with a certain filter shape smooths the contour, breaks narrow lines, eliminates small islands and sharpens peaks on objects. The operation consists of two parts : first an erosion and secondly a dilation. Equation (4,9) shows the mathematical expression for opening an image A with a certain filter B_{fil}.

$$A \circ B_{fil} = (A \ominus B_{max}) \oplus B_{min} \qquad (4,9)$$

In this equation the templates B_{min} and B_{max} have the same amount of do care pixels and are arranged in the same shape. In the first part of the operation ($A \ominus B_{max}$) the black noise is eroded (removed) and in the second part the dilation operation restores the original size of the objects. Figure 4.11 shows a theoretical example of the opening operation.

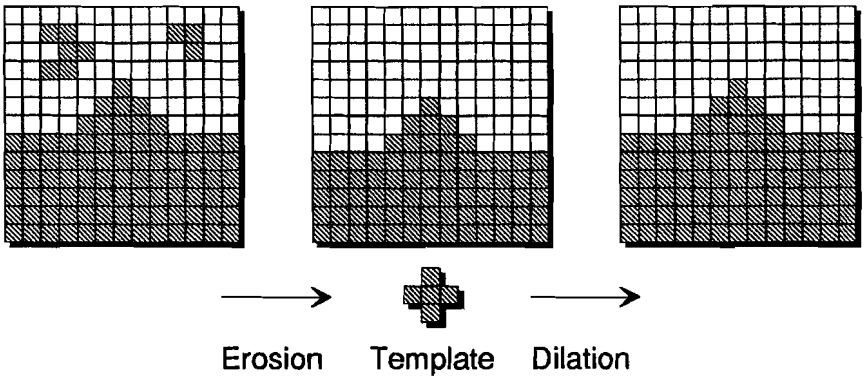


Figure 4.11 *Removing white noise with opening operation*

As can be seen in figure 4.11 the shape of the window is again very important. The filter operation changes the size of the objects in the image as described in paragraph 4.2.4 if the wrong shape is selected.

4.3.2 Closing

The opposite of the opening operation is the closing operation. Closing an image with a certain filter shape smooths the contour, fills narrow breaks in small lines, eliminates small holes and fills gaps on the contour. The operation exists of two parts : first a dilation and secondly an erosion. Equation (4,10) defines the closing of image A with filter B_{fil}.

$$A \bullet B_{fil} = (A \oplus B_{min}) \ominus B_{max} \qquad (4,10)$$

The first part dilates (removes) the black noise with the template B_{min} and in the second part the erosion operation (B_{max}) restores the original size of the objects (see figure 4.12).

The two noise filter techniques opening and closing can be build with the same configurations and templates as the erosion and dilation operations (see paragraph 4.2.2). Only now the result of the first operation is the input of the second operation.

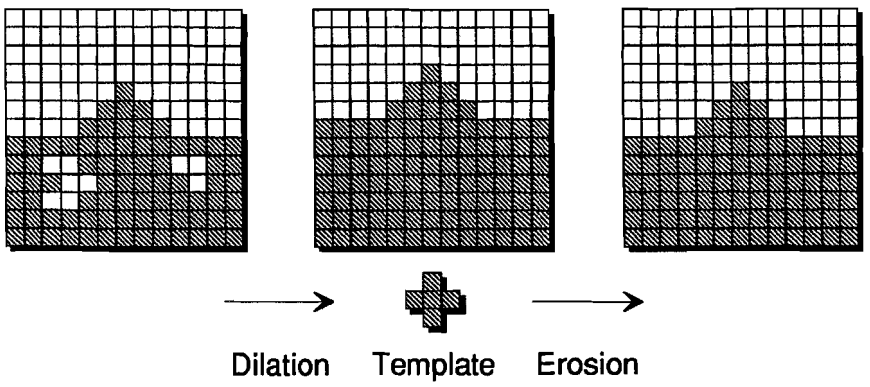


Figure 4.12 *Removing white holes (noise) with closing operation*

This cascade principle is implemented with two memories. In case of a closing operation the input image will be dilated first and the result will be stored in Memory 1. Next the OPTIC will be reprogrammed (selecting an other output) and the image in Memory 1 will be eroded. The final result will be stored in Memory 2. This so called PingPong process is used when only one processor is available (see figure 4.13).

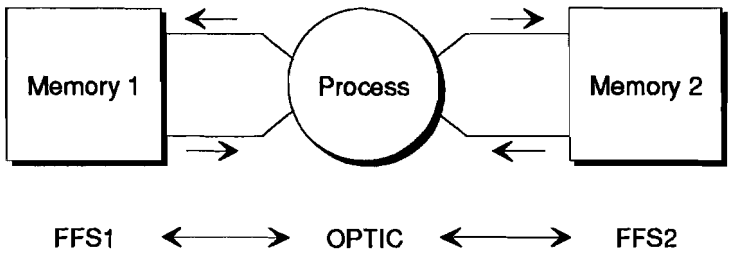


Figure 4.13 *Implementation of the Pingpong process*

With the above mentioned process it is possible to construct a more complex filter. Figure 4.14, 4.15 and 4.16 show the results of a combination of both the opening and the closing operation. Before the input image from the camera will go to the PAPS system and the OPTIC it will pass through a Single Board Image Processor (SBIP). The output image from this SBIP contains both white and black noise. This noise is generated with a random generator in the SBIP and is added to the input image.

The advantage of this system is that it is possible to control the colour and the size of the noise in the image. In this way it is possible to test the filter performance of the different filter shapes. Depending on the selected shape (amount of do care pixels) it is possible to change the size of the noise. As expected removes a 5x5 matrix filter larger noise peaks as a 3x3 filter.

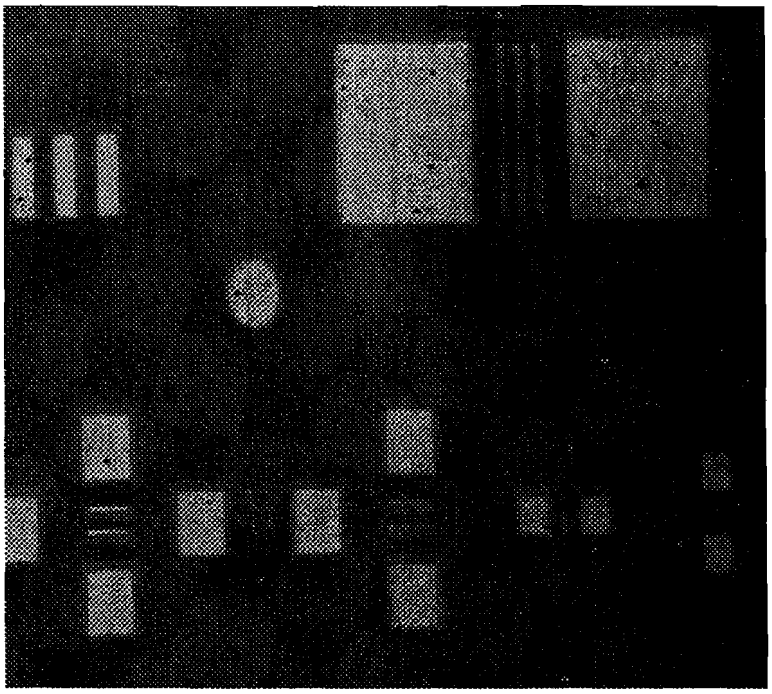


Figure 4.14 *Image with both white and black noise*

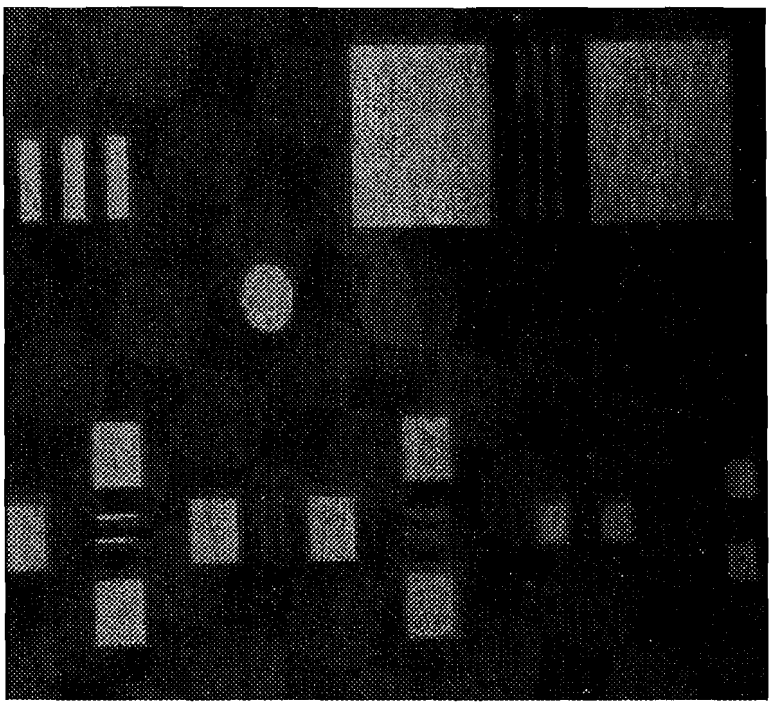


Figure 4.15 *Black noise removed with opening operation*

Figure 4.15 shows the result of the first operation. In this case the black noise is removed with the opening operation. Figure 4.16 gives the final result of the filter operation. In this figure the white noise is removed with the closing operation.

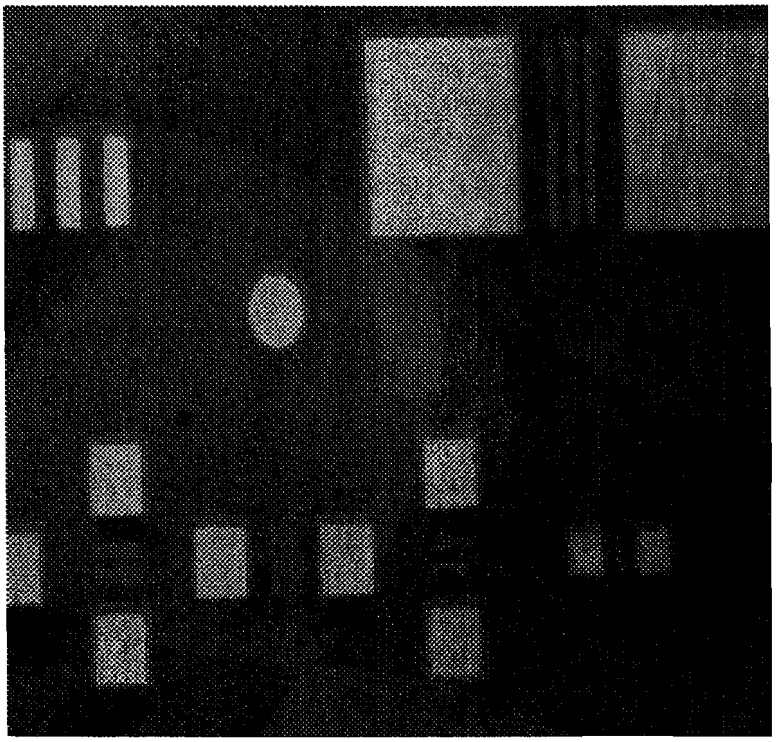


Figure 4.16 *White noise removed with closing operation*

The above described operation is called noise filtering and is expressed in equation (4,11).

$$\begin{aligned} \textit{Noise Filter} &= (A \circ B_{fil}) \bullet B_{fil} \\ &= (((A \ominus B_{max}) \oplus B_{min}) \oplus B_{max}) \ominus B_{min} \end{aligned} \qquad (4,11)$$

The filter is a combination of the pixel processing techniques erosion and dilation and therefore a combination of the basic functions of the OPTIC : minimum and maximum sorting.

4.3.3 Edge preserving filter

A property of the noise filter techniques described above is the fact that small lines will be removed from the image. In some cases, especially inspection applications, where cracks of a certain length should be detected this can be a disadvantage. As a result of this problem an edge preserving filter has been developed. This filter removes noise from an image but keeps small lines unchanged. The

operation uses a filter with a shape like a ring. The algorithm calculates the maximum and minimum of the pixels on the ring and compares the difference (*Max-Min*) with a programmable *Contrast_Threshold*. If this (*Max-Min*) is smaller than the threshold the *CenterPixel* is replaced with the average of the template. This results in the removing of the noise when the "ring" fits around the noise. A small line in the image is not affected by the filter because this line will cross the ring and the difference (*Max-Min*) will be larger than the threshold. In this case the *CenterPixel* is not replaced (see figure 4.17).

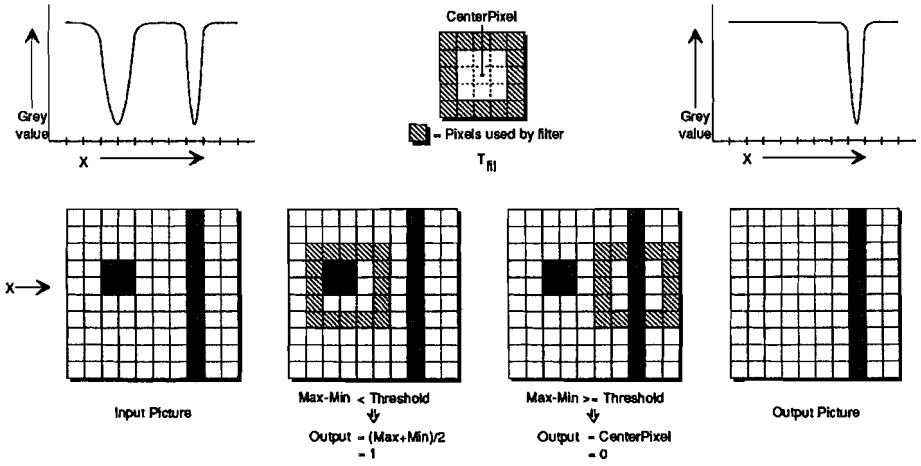


Figure 4.17 *Example of edge preserving filtering on an image*

This edge preserving filter operation can be written as follows :

if (*Max-Min*) ≥ *Contrast_Threshold*
{
    Output = *CenterPixel*
else
{
    Output = ((*Max+Min*)/2)
}

If the *CenterPixel* (CP) of the "ring" moves along the pixels belonging to the spike, the noise will only be removed if the "ring" fits around the noise in these positions (see figure 4.18). So the size of the "ring" filter determines which noise spikes will be removed from the image.

A disadvantage of a larger "ring" size is that the distance between the noise spikes has to be larger before the noise will be removed. If the noise is too close to an other edge (a noise spike or a object), the "ring" (R) will also fit around this next noise spike or edge (see figure 4.19). In this case the noise is not removed.

The minimum distance (D_{min}) between a noise spike and an other edge can be calculated with the relation in equation (4,12)

$$D_{min} = (\#\ pixels\ between\ R\ ,\ CP) + 1 \qquad (4,12)$$

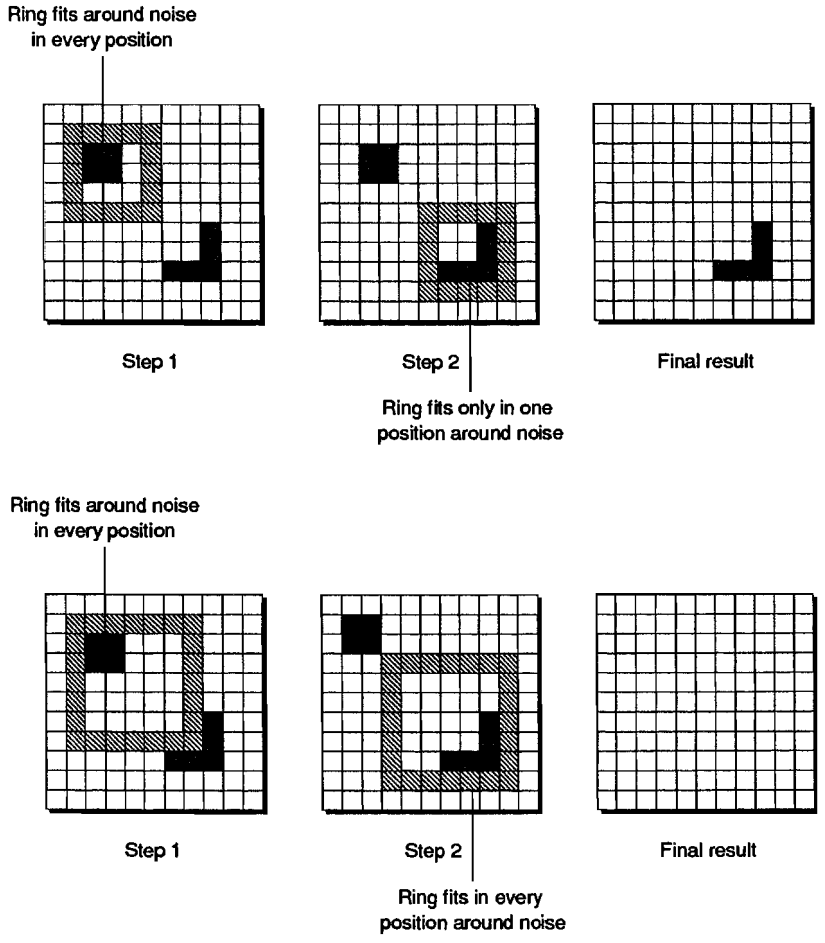


Figure 4.18 *Effect of the "ring" size on the filter result*

The edge preserving technique is tested with the configuration and window as listed in table 4.3. As shown in the table the size of the ring used is 5x5 pixels. With this filter it should be possible to remove all the noise of 2x2 pixels or smaller. This was tested with the image from figure 4.20. The input image passes first through an SBIP before going to the PAPS system and the OPTIC . The SBIP adds noise to the image. With the SBIP the size and shape of the noise can be controlled. The result is an image with user defined noise. Only the position of this noise is at random.

The result of the filter (OPTIC output DO1) can be seen in figure 4.21. It is clear that the noise is removed by the filter. So it is proved that the ring of 5x5 pixels is able to remove all the noise smaller

or equal than 2x2 pixels. Only the larger noise remained in the image. This was tested with a cursor moving along the image. With this tool the size of the remaining noise was measured. During this measurement also some noise of size 2x2 was found. These noise spikes were not removed by the filter because they were to close to an edge or a line. This can also be seen in figure 4.21.

Both the input and output image from the process where stored in the SBIP for further examination. To check if the edges and the small lines have been influenced by the filter a plot of a row is shown in the upper part of both the images. The plot shows the greyvalues of every pixel along the row.

Table 4.3 *Example of the implementation of the edge preserving noise filter*

Template pixel attributes								Configuration	Description
x	x	x	x	x	x	x	x	Window shape	8x8
x	**f**	**f**	**f**	**f**	**f**	x	x	NMS	Don't care
x	**f**	x	x	x	**f**	x	x	CPNF	Noise filter
x	**f**	x	CP	x	**f**	x	x	MODE	DO0 = (Max+Min)/2
x	**f**	x	x	x	**f**	x	x		DO1 = CPNF
x	**f**	**f**	**f**	**f**	**f**	x	x	Threshold	Contrast_threshold = 20
x	x	x	x	x	x	x	x	DIFF	Max-Min
x	x	x	x	x	x	x	x	TH0	Don't care
								TH1	Don't care

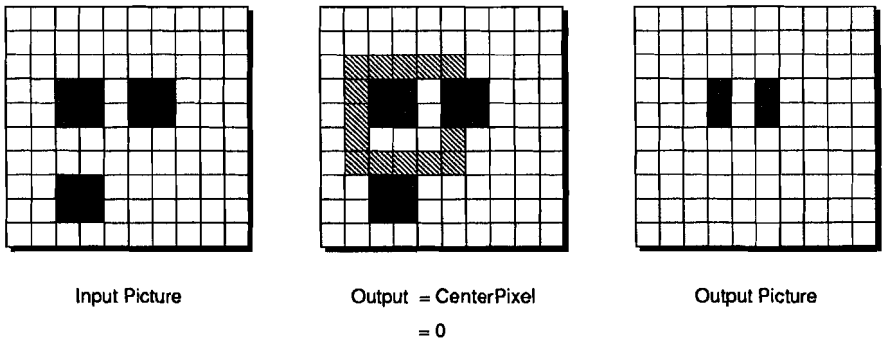


Figure 4.19 *The effect of the distance between the noise on the result*

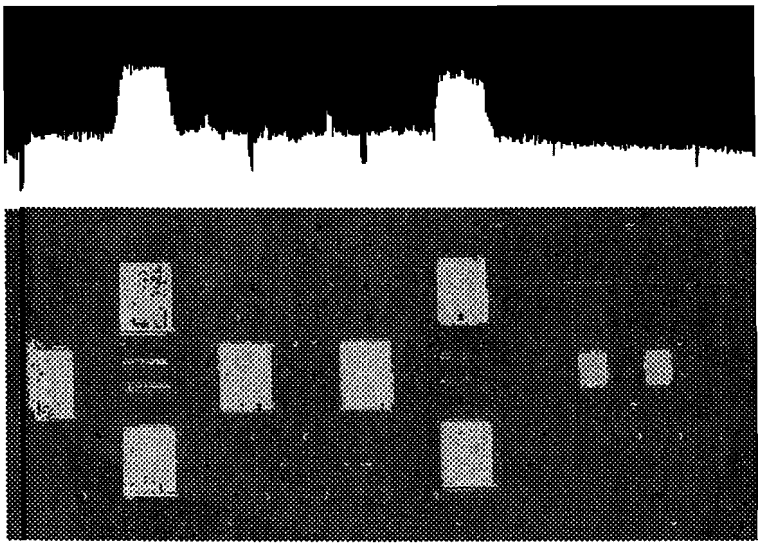


Figure 4.20 *Image before filtering. The image contains noise and a vertical line*

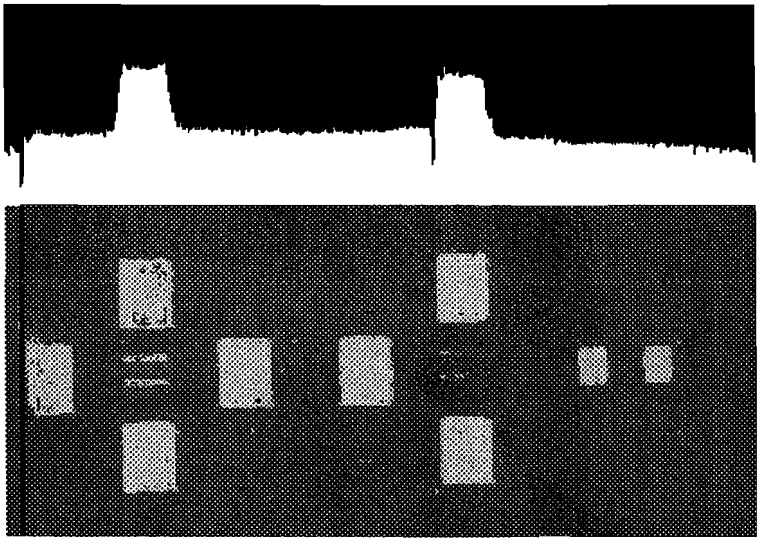


Figure 4.21 *Image after filtering. The noise is removed but the line is unchanged*

The remaining noise spikes in figure 4.21 are the result of the effects as discussed in this paragraph (noise to close to a object or ring size to small).

4.4 Dynamic thresholding

Thresholding is used to convert a 8-bit greyvalue picture to a binary (black and white) one. In some applications a global threshold is used but sometimes it is not possible to set just one global threshold which is valid for every position in one image. In dynamic thresholding the local mean of a window of pixels is calculated. This mean value ((*Max+Min*)/2) is compared with the actual value of the *CenterPixel*. If this value is larger or equal the result is a white pixel, otherwise a black pixel is substituted for the *CenterPixel* position. In order to reduce the influence of noise on the result a *Contrast_Threshold* is selected. The thresholding operation is disabled unless the local (*Max-Min*) value is greater than the programmed *Contrast_Threshold*.

The above mentioned dynamic threshold operation can be written as follows :

```
if (Max-Min) ≥ Contrast_Threshold
{
        if CenterPixel ≥ ((Max+Min)/2)
                Output = 1
        else
                Output = 0
}
else
{
        Output = 0
}
```

Figure 4.22 gives an example of a dynamic threshold operation. The result of the thresholding is that only the local edges with *Max-Min* ≥ *Contrast_Threshold* inside the template are detected.

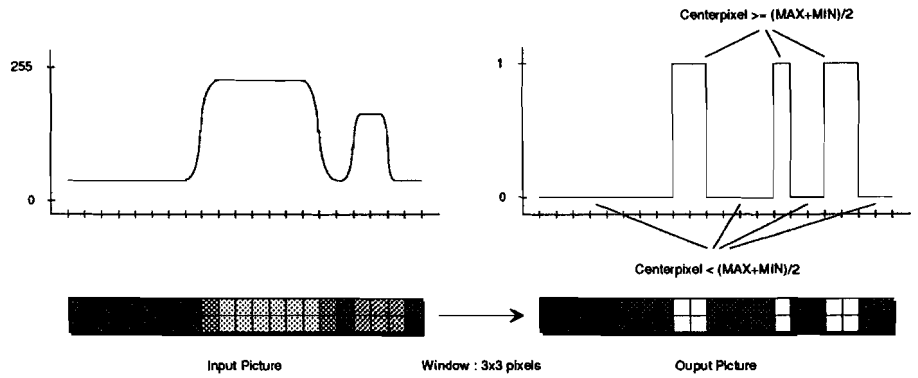


Figure 4.22 *Dynamic thresholding on a greylevel image*

The dynamic threshold function is implemented in the OPTIC with the configuration and the template given in table 4.4. Figure 4.23 gives the input image. Figure 4.24 shows the dynamic threshold operation. The resulting image is formed with the binary output TH1 from the OPTIC.

Table 4.4 *Example of the implementation of the dynamic threshold operation*

Template pixel attributes								Configuration	Description
x	x	x	x	x	x	x	x	Window shape	8x8
x	x	x	x	x	x	x	x	NMS	Dynamic Thresholding
x	x	f	f	f	x	x	x	CPNF	Centerpixel
x	x	f	f	f	x	x	x	MODE	DO0 = DIFF
x	x	f	f	f	x	x	x		DO1 = (Max+Min)/2
x	x	x	x	x	x	x	x	Threshold	Contrast_threshold = 10
x	x	x	x	x	x	x	x	DIFF	Max-Min
x	x	x	x	x	x	x	x	TH0	Don't care
								TH1	Binary output image

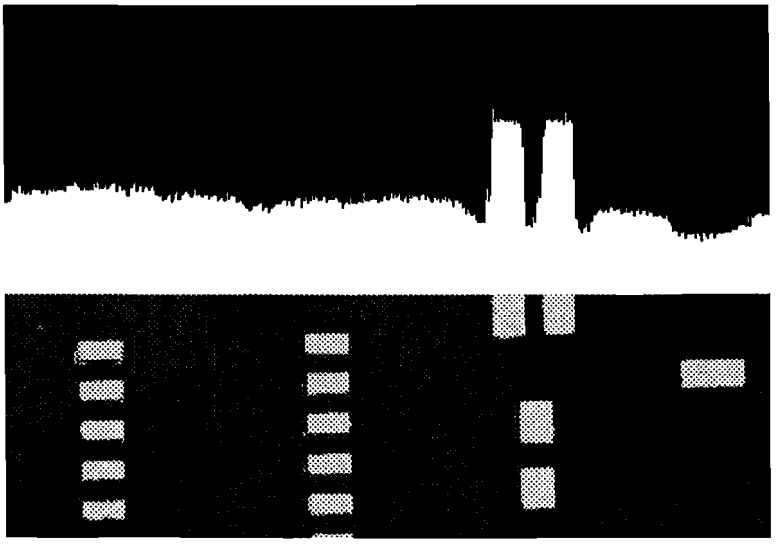


Figure 4.23 *Input picture for the dynamic threshold operation.*

To get an impression of the result of the operation the upper part of the images contain a plot of a row in the image. Therefore both pictures have been stored and processed in the SBIP. The plots show which edges are detected (thresholded) and which are not.

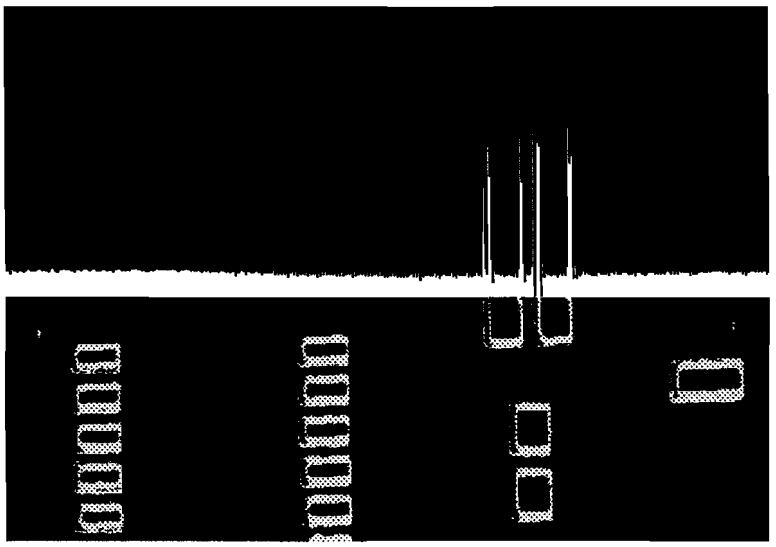


Figure 4.24 *Result after thresholding with a filtersize of 3x3 pixels*

As shown in the examples the dynamic threshold operation can be used for edge detection in an image. It is a very useful operation for detecting local edges (noise spikes). The value of the threshold and the amount of do care pixels in the template determine the outcome of the dynamic threshold operation. The effect of these two variables will be discussed in the next paragraph.

4.4.1 The effect of the contrast threshold on the dynamic operation

If the threshold is too low noise will affect the outcome of dynamic threshold operation. In this case even noise will be detected. If the threshold is too high (higher as the contrast of the objects) the resulting image will be black. Before the threshold will be selected the maximum contrast of the noise within the window needs to be calculated. If the selected threshold is higher then the calculated value, the influence of noise can be neglected.

Figure 4.25 gives an example of the effect of the threshold on the result. The figure shows what happens if the threshold is not selected properly. In one case (*Contrast_Threshold* < *H*) the noise is also thresholded to one and the result is not correct.

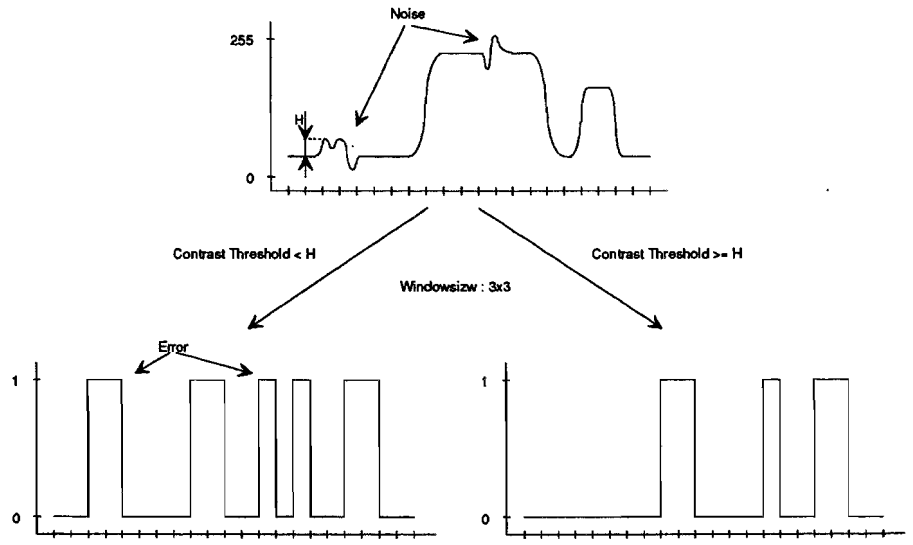


Figure 4.25 *The effect of the threshold on the result of the dynamic threshold operation*

Figure 4.26 gives the result of the dynamic thresholding operation of figure 4.23 with a smaller threshold. The result shows more noise in the output image (see also the plot of a row in the upper part of the image).

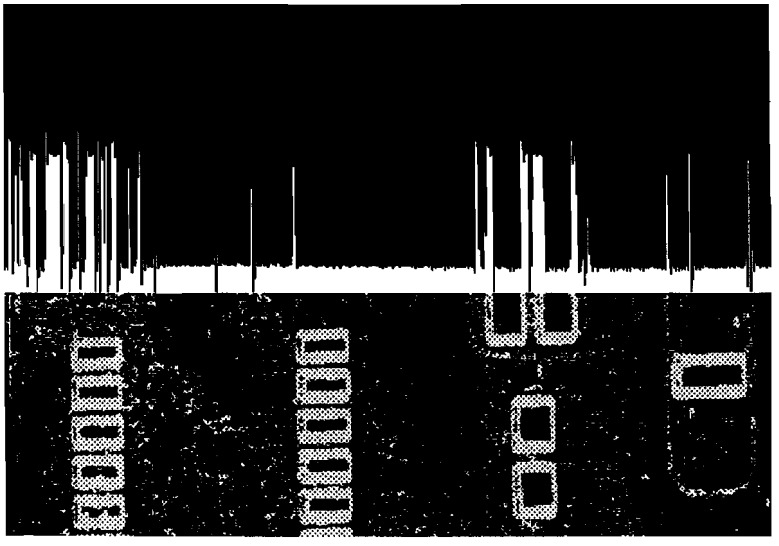


Figure 4.26 *Result of the thresholding with a smaller threshold*

4.4.2 The effect of the template on the dynamic operation

An other way to influence the result of the dynamic threshold operation, is to change the amount of do care pixels in the template. If this amount is increased, more pixels are used to calculate the contrast inside the template. In this case even objects with smooth edges (the contrast changes very slow) are being thresholded .

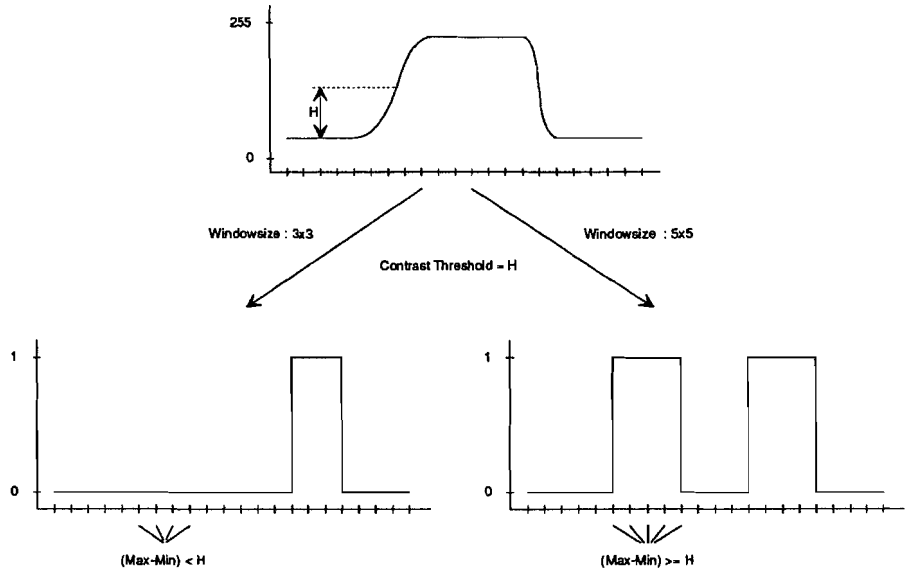


Figure 4.27 *The effect of the window size on the dynamic threshold operation*

Figure 4.27 gives a theoretical example of the effect of the template on the result. The figure shows what happens if the amount of do care pixels is not selected properly. With a shape of 3x3 pixels the left edge of the object is not detected and thresholded. If a larger shape (5x5) is used, the edge is detected and the right detected edge becomes thicker. The same effect can be seen in figure 4.28. In this figure the operation is performed with a template of 5x5 do care pixels. The result contains more and thicker edges.

4.5 Shape recognition

The OPTIC is also able to recognize or detect different kinds of objects. It uses two different kinds of recognition techniques : edge detection for detecting every edge of an image and template matching for recognizing user programmable shapes and objects. Both of these techniques use the difference between the maximum and the minimum pixel inside a template. The advantage of this technique is that it makes both types of recognition independent from the light in the image and the focusing of

the camera (local difference between a maximum and minimum remains the same). In the next sub-paragraphs these different pixel processing techniques are explained.

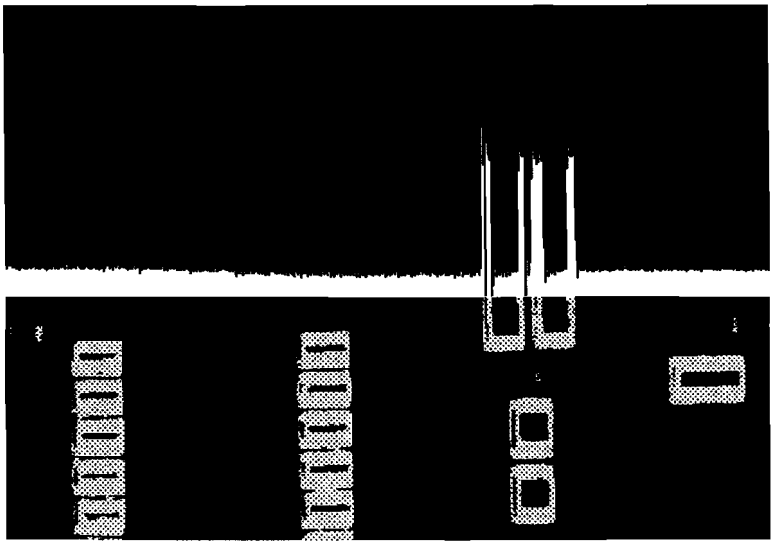


Figure 4.28 *Resulting image after thresholding with a 6x6 do care pixels*

4.5.1 Edge detection

Edge detection is a technique used to detect every edge in an image. It uses the erosion and dilation operation to find the edges. When using this operation the image will be at the same time eroded with B_{max} and dilated with B_{min}. The result of two operations are pixel-wise subtracted. This difference is displayed in the output image (see the relation in equation (4,13)).

$$I_{Edge\ Detection} = I_{Dilation} - I_{Erosion} = (A \oplus B_{min}) - (A \ominus B_{max}) \quad (4,13)$$

Figure 4.29 gives an example of the edge detection operation. The upper part of the figure shows the eroded object and the lower part contains the dilated object. In the next step the two images are subtracted. The with of the detected edge depends on the amount of pixels eroded or dilated from the image (size and shape of B_{max} and B_{min}).

This edge detection technique can be build with the OPTIC configurated as listed in table 4.5. The result of the detection can be displayed with the DO0 (DIFF) output or with the binary output TH0.

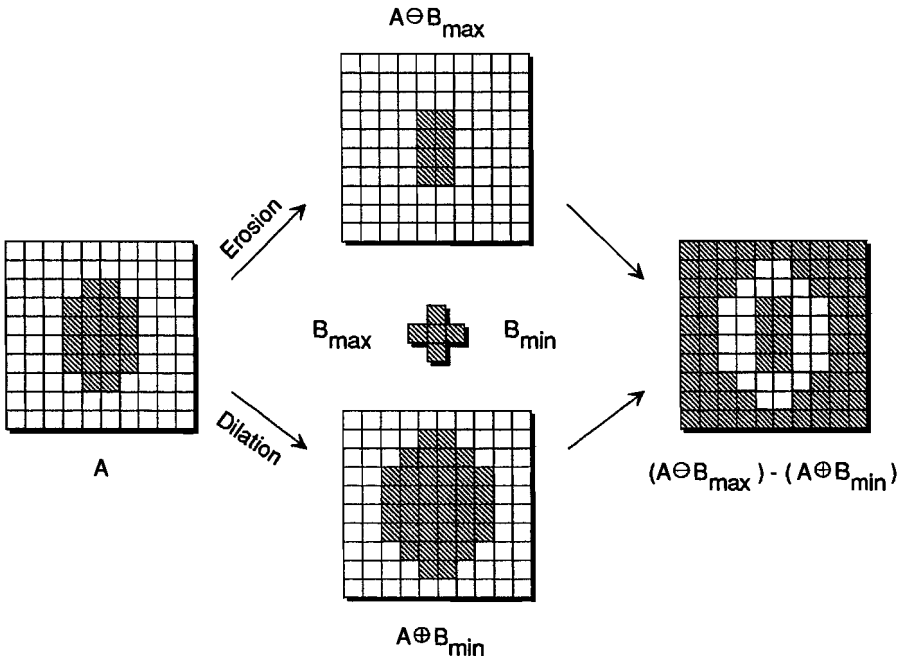


Figure 4.29 *Example of the edge detection operation on an image*

The binary image is the result of a threshold operation performed on the calculated DIFF (Max-Min) and can be written as follows:

```
if (Max-Min) ≥ Threshold
{
        Output = 1
}
else
{
        Output = 0
}
```

Figure 4.30 and figure 4.31 show the edge detection operation.

Table 4.5 *Example of the implementation of the edge detection operation*

Template pixel attributes								Configuration	Description
x	x	x	x	x	x	x	x	Window shape	8x8
x	x	x	x	x	x	x	x	NMS	Don't care
x	x	x	x	x	x	x	x	CPNF	Don't care
x	x	x	x	x	x	x	x	MODE	DO0 = DIFF
x	x	x	x	x	x	x	x		DO1 = don't care
x	x	x	x	x	**f**	**f**	**f**	Threshold	Threshold
x	x	x	x	x	**f**	**f**	**f**	DIFF	Max-Min
x	x	x	x	x	**f**	**f**	**f**	TH0	DIFF ≥ Threshold
								TH1	Don't care

Depending on the application the greylevel edge output (DO0) or the binary edge output (THO) can be displayed.

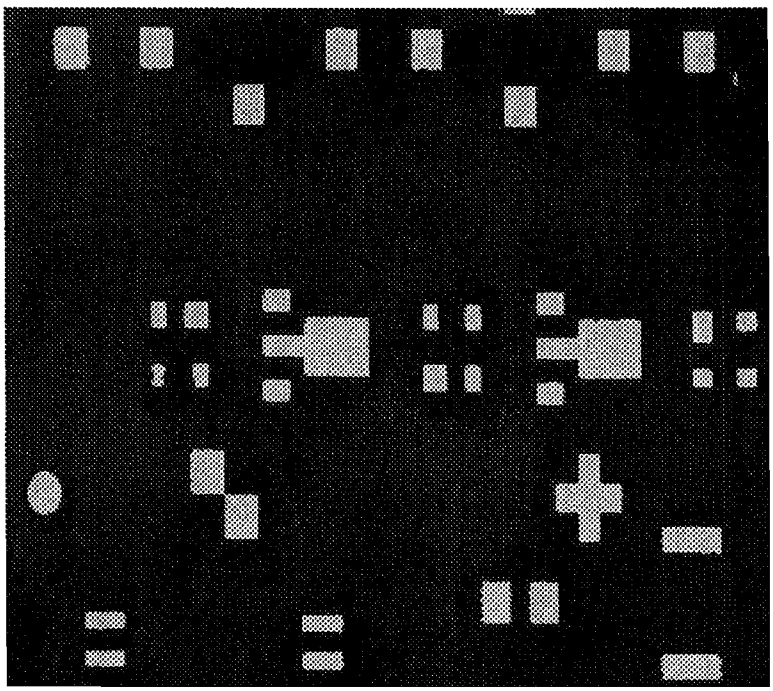


Figure 4.30 *Image before edge detection*

4.5.2 Greylevel template matching

Template matching is a very powerful operation for object recognition. However its main disadvantage is that it usually requires a binary image as input. With the sorters implemented in the OPTIC it is possible to perform binary template matching on greylevel images. This Dynamic Range Correlation

(DRC) operation uses a template which contains light, dark and don't care pixels. The DRC calculates the minimum value of the light pixels in the template at which the template would match if it had been used in a binary-template matcher. Also the maximum value of the dark pixels for a perfect match in a binary template-matcher is being calculated. If the number of possible 'matching-threshold-values' is one or more, a match has been found.

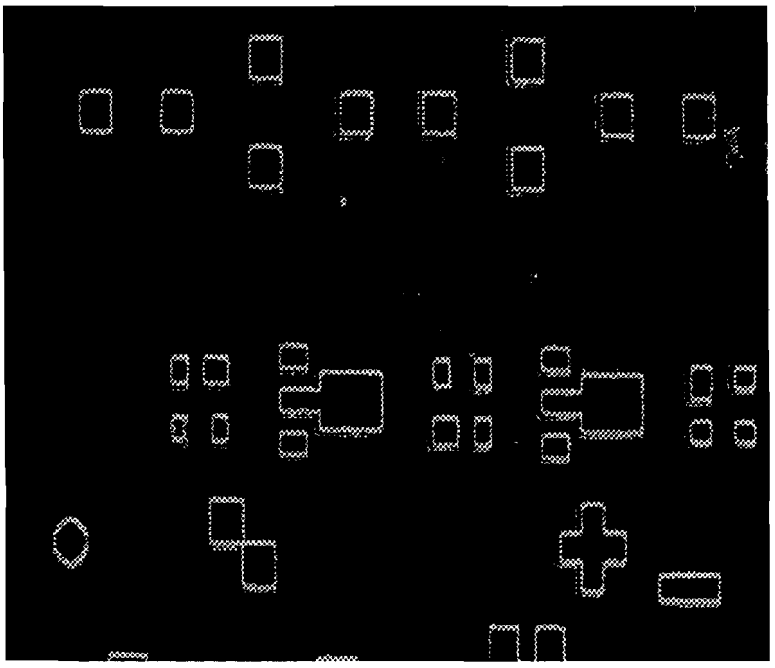


Figure 4.31 *Resulting image after edge detection with 3x3 pixels*

This algorithm invented by Bernsen [3] can be written as follows :

Range = [Min(light pixels) - Max(dark pixels)]
if Range < 0
{
    Range = 0
}
if Range ≥ Threshold
{
    Output = 1
}
else
{
    Output = 0
}

Next to the binary output it is also possible to display the *Range*. The output will be a greylevel image. Figure 4.32 gives an example of the DRC operation. With the template used in this figure a rising edge can be detected.

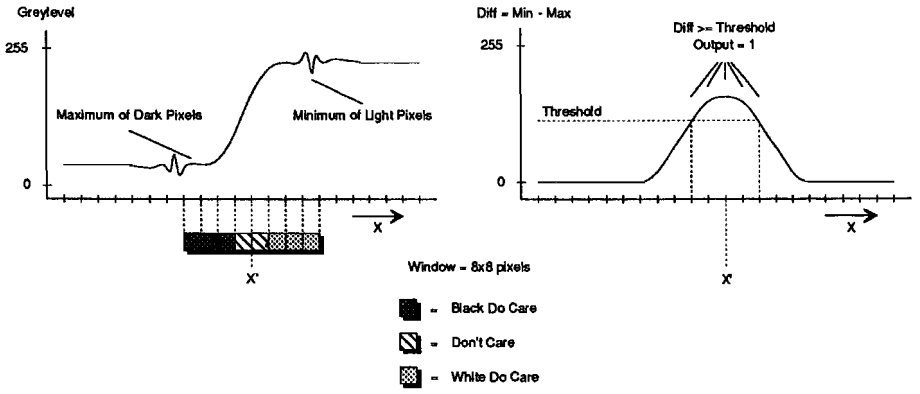


Figure 4.32 *Example of template matching on a greylevel image*

The DRC operation will make errors if the noise level inside the template close to the object has almost the same size as the edge (imaging that the "dark" noise in figure 4.32 is higher). In this case the *Max(dark pixels)* will be almost the same as the *Min(light pixels)*. As a result of this the *Range* will be small or even negative and the edge is not recognized.

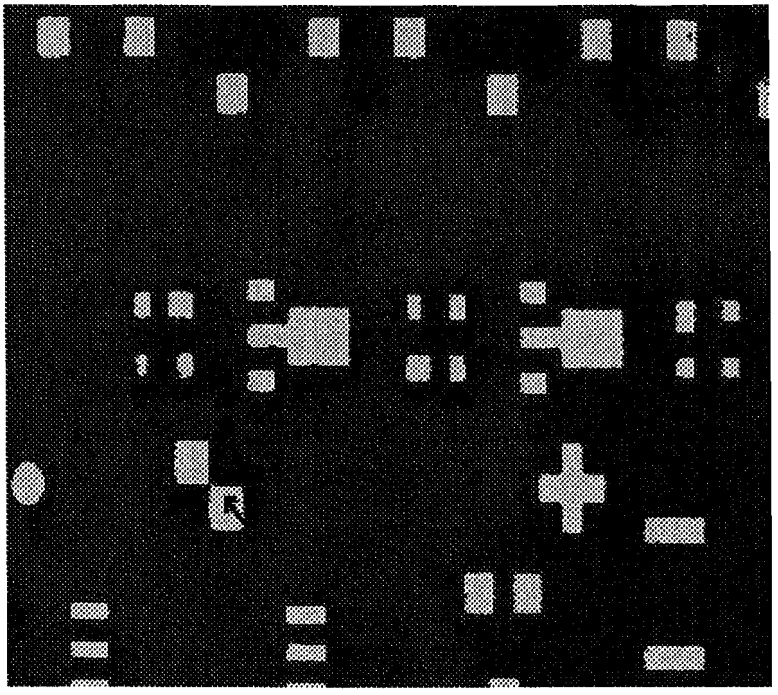


Figure 4.33 *Example of the DRC operation*

The greyvalue template matching technique can be implemented in the OPTIC with the configuration listed in table 4.6. In this case the *Range* is represented by DIFF. The template used is also listed in this table. With this template it is possible to recognize the two close connected squares in figure 4.33.

Table 4.6 *Example of the implementation of greylevel template matching*

Template pixel attributes								Configuration	Description
1	1	1	x	x	0	0	0	Window shape	8x8
1	1	1	x	x	0	0	0	NMS	Don't care
1	1	1	x	x	0	0	0	CPNF	Don't care
1	1	1	x	x	0	0	0	MODE	DO0 = DIFF
x	x	x	x	x	x	x	x		DO1 = Don't care
0	0	0	x	x	1	1	1	Threshold	Threshold = 12
0	0	0	x	x	1	1	1	DIFF	Min-Max
0	0	0	x	x	1	1	1	TH0	DIFF ≥ Threshold
								TH1	Don't care

The don't care pixels in the template (x in table 4.6) allow some rotation between the object and the template. The result of these don't cares will be that the template matches on more than one position with the object in the image. Changing the *Threshold* can overcome this problem in most of the cases. An other solution to this problem is given in the next sub-paragraph.

The binary output of the OPTIC (TH0 = 1 if DIFF > *Threshold*) is send to a special PAPS card. This card places an arrow in the image if the recognition signal (TH0) is high. Figure 4.33 shows the result of the template matcher (and the PAPS-card) combined with the original input image. In the figure the arrow marks the place of all the recognitions.

4.5.3 Recognition of larger shapes

With the window shapes available in the OPTIC it is only possible to recognize shapes or objects of 1x64, 2x32, 4x16 and 8x8 pixels. If it is necessary to recognize larger shapes the input image needs to be subsampled in both horizontal and vertical direction. A subsampling of Hor_{sub} in horizontal direction means that only one pixel will pass to the OPTIC every Hor_{sub} clock cycles. A subsampling of Ver_{sub} in vertical direction means that of every Ver_{sub} lines only one will pass to the linebuffer.

The subsampling technique can be used to recognize large shapes in the image. Table 4.7 shows the template used to recognize the objects in the image. Without subsampling this template will only recognize white objects of approximately 6x6 pixels on a black background.

With a subsampling of Hor_{sub},Ver_{sub} the template can recognize objects of approximately 6xHor_{sub}x6xVer_{sub} pixels. Subsampling "enlarges" the size of the template used. The new template size can be calculated with the relation in equation (4,14).

$$\begin{aligned} \Delta W_{x_{new}} &= \Delta W_{x_{old}} \cdot Hor_{sub} \\ \Delta W_{y_{new}} &= \Delta W_{y_{old}} \cdot Ver_{sub} \end{aligned} \qquad (4,14)$$

The result of the template matching with a subsampling of 1,1 (normal) is given in figure 4.34. Figure 4.35 and 4.36 show the result if the sampling is changed from 1,1 to 2,2 and 3,3.

Table 4.7 *Template used for template matching*

Template pixel attributes							
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	x	x	x	x	x	x	**0**
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**
0	**1**	**1**	**1**	**1**	**1**	**1**	**0**
0	x	x	x	x	x	x	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

As can be seen in the three figures the objects (pointed to by the arrows) recognized change when the subsampling is changed. The larger the subsampling the larger the objects recognized. This is in harmony with the expected result.

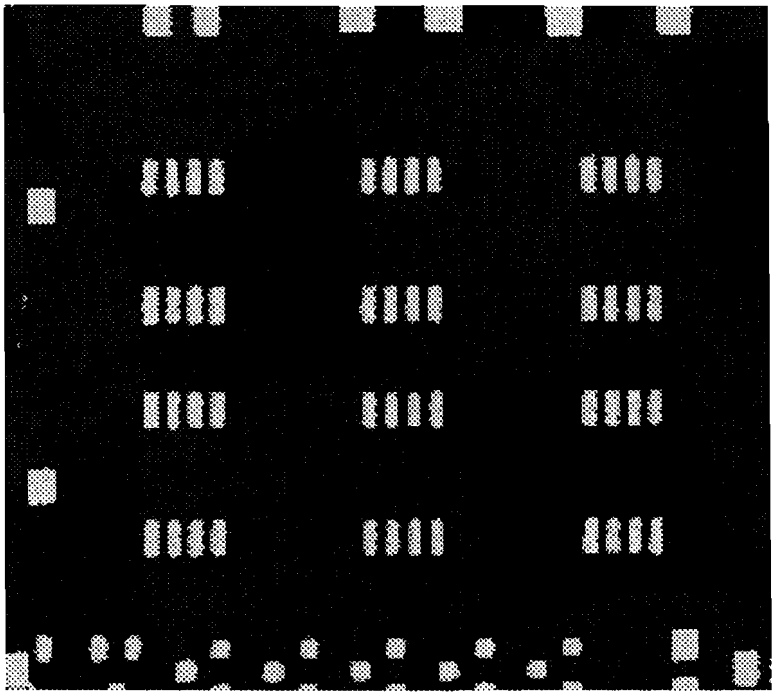


Figure 4.34 *Result of the template matching with Hor_{sub}=1 and Ver_{sub}=1*

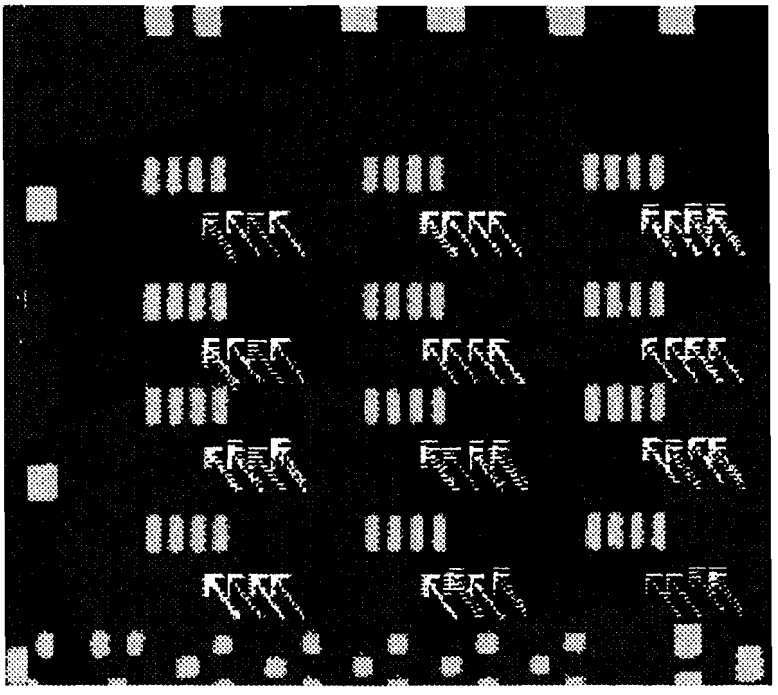


Figure 4.35 *Result of the template matching with Hor_{sub}=2 and Ver_{sub}=2*

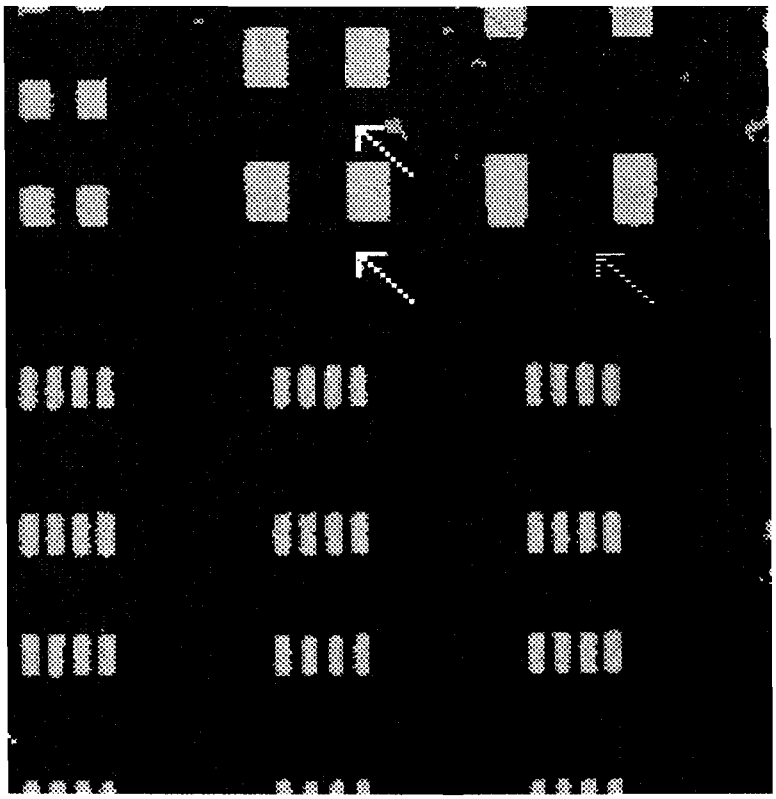


Figure 4.36 *Result of the template matching with Hor_{sub}=3 and Ver_{sub}=3*

4.5.4 Non Maximum Suppression

To improve the DRC result when using the filter for edge detection the Non-Maximum Suppression technique can be used. Here the DRC result is the input to a second filter. This filter works with a template of do-care pixels which are on a line through the *CenterPixel* and orthogonal to the edge programmed in the first filter (see figure 4.37). The algorithm used is defined as follow :

```
if (Max = CenterPixel) and (CenterPixel ≠ 0)
{
        Output = 1
}
else
{
        Output = 0
}
```

The output (TH1) of the NMS operation is a binary image. This contains the exact position of the shape recognized in the template matcher.

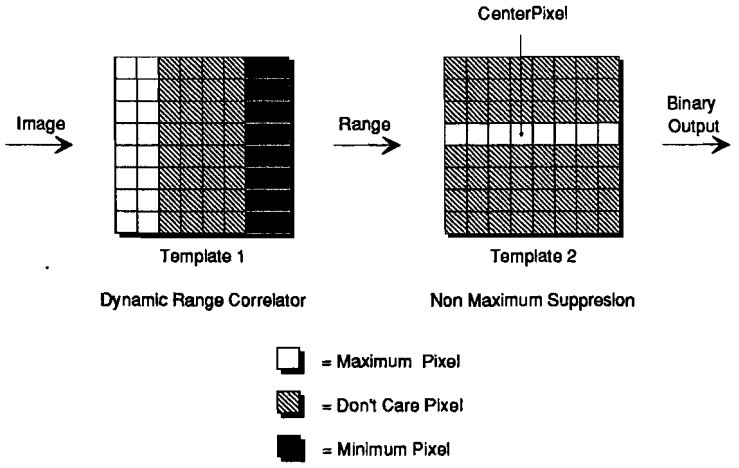


Figure 4.37 *Implementation of the NMS operation*

The above algorithm is illustrated in figure 4.38.

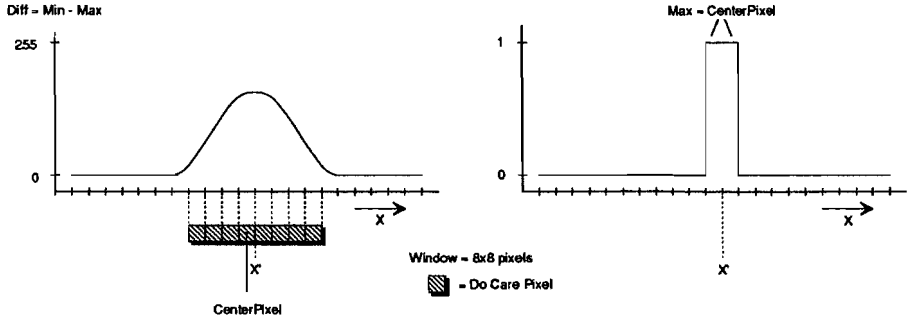


Figure 4.38 *Example of template matching with NMS*

The disadvantage of the above algorithm is the fact that it is very sensitive to noise in the output of the template matcher. Noise in this output will be detected as soon as the *CenterPixel* of the second template matches on this noise. So the output of the NMS will contain a lot of noise (see also figure 4.39).

The result of the NMS operation can be improved by placing a comparator and a multiplexer between the template matcher and the NMS operation. In this part the output of the matcher (*Range*) will be compared with a *Contrast_threshold.* Only a *Range* > *Threshold* will go to the second stage (NMS) (see figure 4.40).

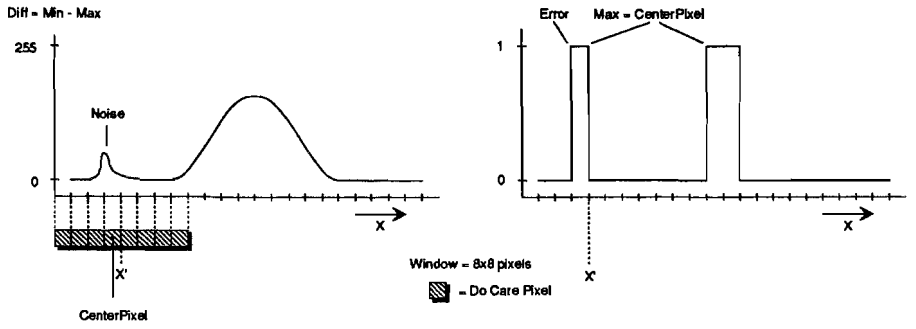


Figure 4.39 *Effect of noise on the NMS operation*

As a result of this hardware the noise in the output of the matcher won't influence the NMS operation. The NMS operation will then work as illustrated in figure 4.38.

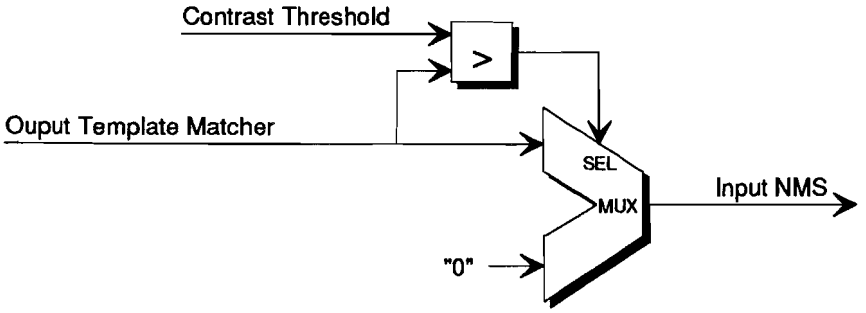


Figure 4.40 *Hardware to improve the NMS operation*

The "old" NMS operation was tested with the configuration and template as listed in table 4.8. The first part of the operation (DRC) was performed with the template as given in figure 4.37 and the configuration as listed in table 4.6. The two separate operation where performed in a PingPong process as described in sub-paragraph 4.3.2.

The output result from the NMS operation contained a lot of noise as expected. This noise was the result of the above mentioned process. The result of this NMS operation will only be suitable if the output of the first stage (template matcher) contains no noise which can be achieved by thresholding the *Range* output of the template matcher, as proposed above

Table 4.8 *Example of the implementation of NMS operation*

Template pixel attributes								Configuration	Description
x	x	x	x	x	x	x	x	Window shape	8x8
x	x	x	x	x	x	x	x	NMS	NMS
x	x	x	x	x	x	x	x	CPNF	Don't care
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	MODE	DO0 = Max
x	x	x	x	x	x	x	x		DO1 = Don't care
x	x	x	x	x	x	x	x	Threshold	Don't care
x	x	x	x	x	x	x	x	DIFF	Don't care
x	x	x	x	x	x	x	x	TH0	Don't care
								TH1	Binary NMS output

5 CONCLUSIONS

After building and testing the OPTIC testcard the application research started. During the research a test program and a demo program have been written. With these two programs it is now possible to demonstrate most of the applications of the pixel processing chip OPTIC. Thanks to the research it is now clear that the OPTIC can perform real time pixel processing techniques on greylevel based images. With an OPTIC it is possible to implement some types of filters and recognition techniques. The following on morphological operations based applications possibilities can be performed with the OPTIC in real time:

- Erosion and dilation
- Edge preserving filtering
- Dynamic thresholding
- Edge detection
- Greylevel template matching

When using the OPTIC in a cascade structure it is also possible to perform some extra operations like:

- Opening and Closing
- Noise filtering
- Template matching with Non Maximum Suppression (NMS)

All but one application worked as expected. Only the NMS operation needed some improvement.

As a result of the fact that the OPTIC can perform morphological operations on greyvalue images the operations are more or less independent of the amount of the available light and the focusing of the camera. This and the fact that the OPTIC can perform the pixel processing techniques real time make the chip very useful for different kind of filter and recognition applications.

LITERATURE

[1] *A pipelined MAX-MIN filter image processor*, Silicon & Software Systems Dublin Ireland, September 1990, Doc. nr. 88i06_n_1

[2] *Using the OPTIC image processor*, Silicon & Software Systems Dublin Ireland, June 1991, Doc. nr. 88i06_n_2

[3] Bernsen ir. J.A.C., *Multi-level Input Binary Template Matching*, Philips Nat. Lab., March 1989, Technical note nr. 075/89

[4] Haralick R.M., Sternberg S.R. and Zhuang X., *Image Analysis Using Mathematical Morphology*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-9, Number 4, July 1987, p 532-550

[5] Ekman O., *Templates for alignment and inspection*, Philips CFT-Automation, December 1990, M. Sc. Thesis of the Royal Institute of Technology Stockholm

[6] Gonzalez R.C. and Wintz P., *Digital image processing*, Addison-Wesley Publishing Company 1987, Second edition

[7] Hoeks W.L.M., *Performance evaluation for thinning algorithms with respect to boundary noise*, Philips CFT, May 1992

[8] *PAPS Picture Acquisition and Image Processing System*, N.V. Philips, 1986, Technical Information Department Machine Factories, MFW 3-87-51xx

APPENDIX A

1 OPTIC Testcard

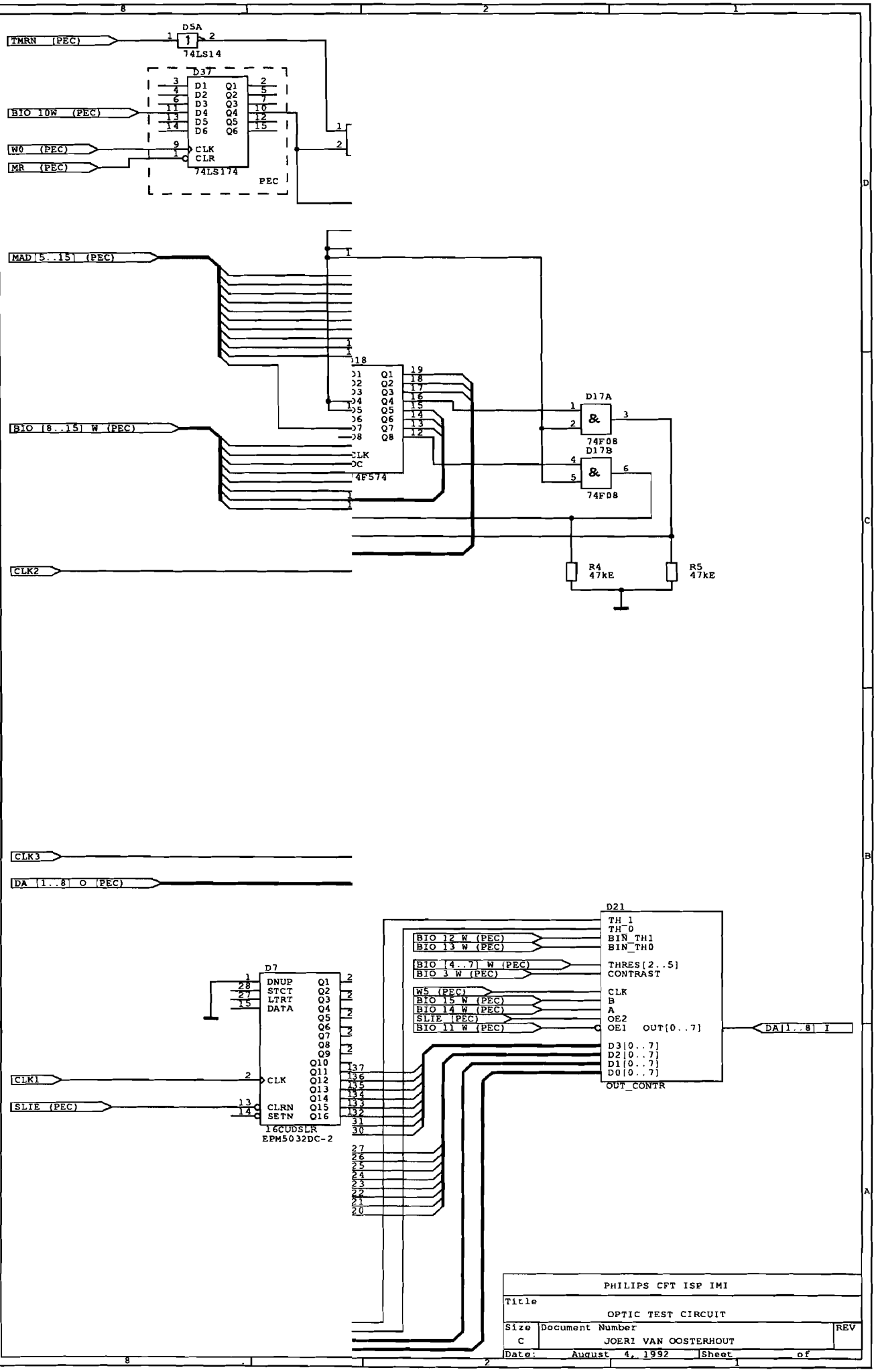
2 Parallel to Serial Converter

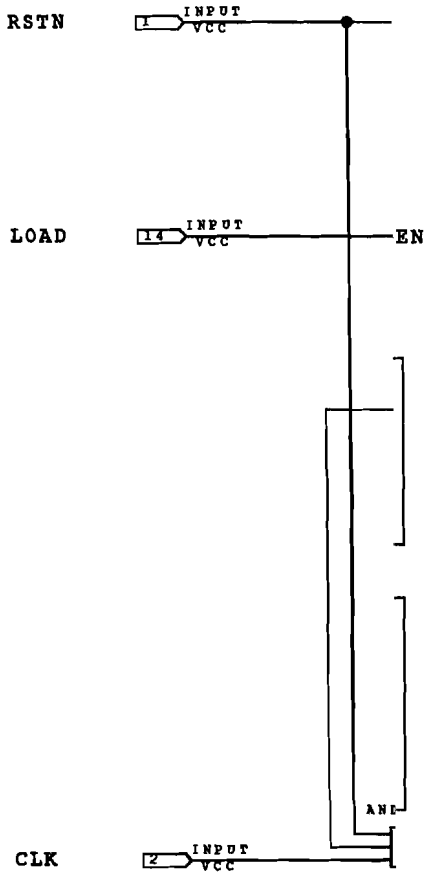
3 Timing circuitry Line Buffer

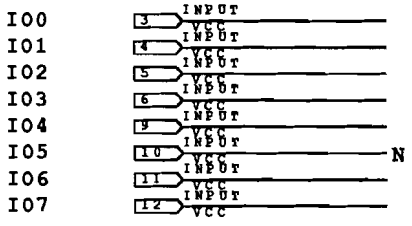
4 Output Control

5 Layout OPTIC Testcard

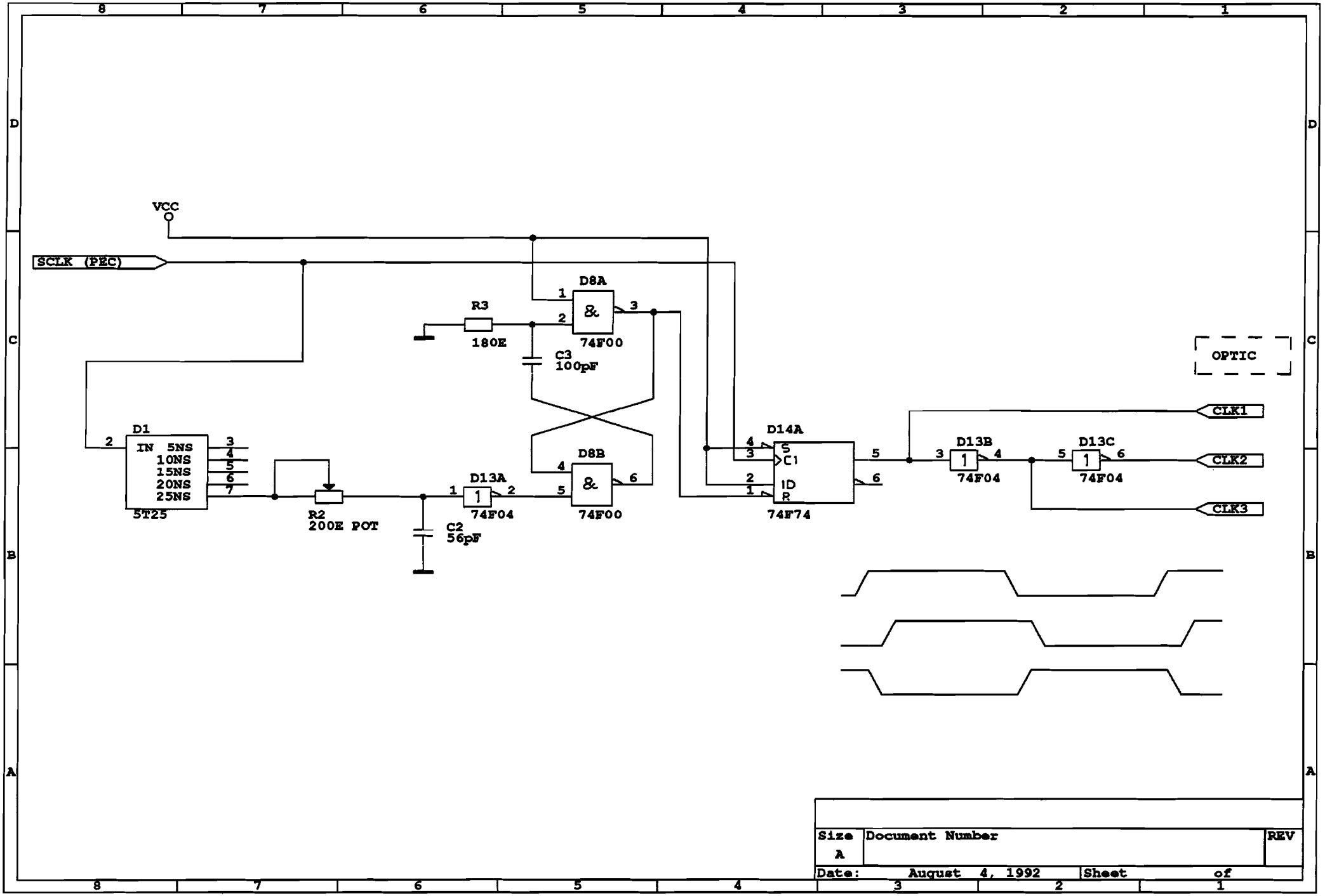
6 Pin layout OPTIC

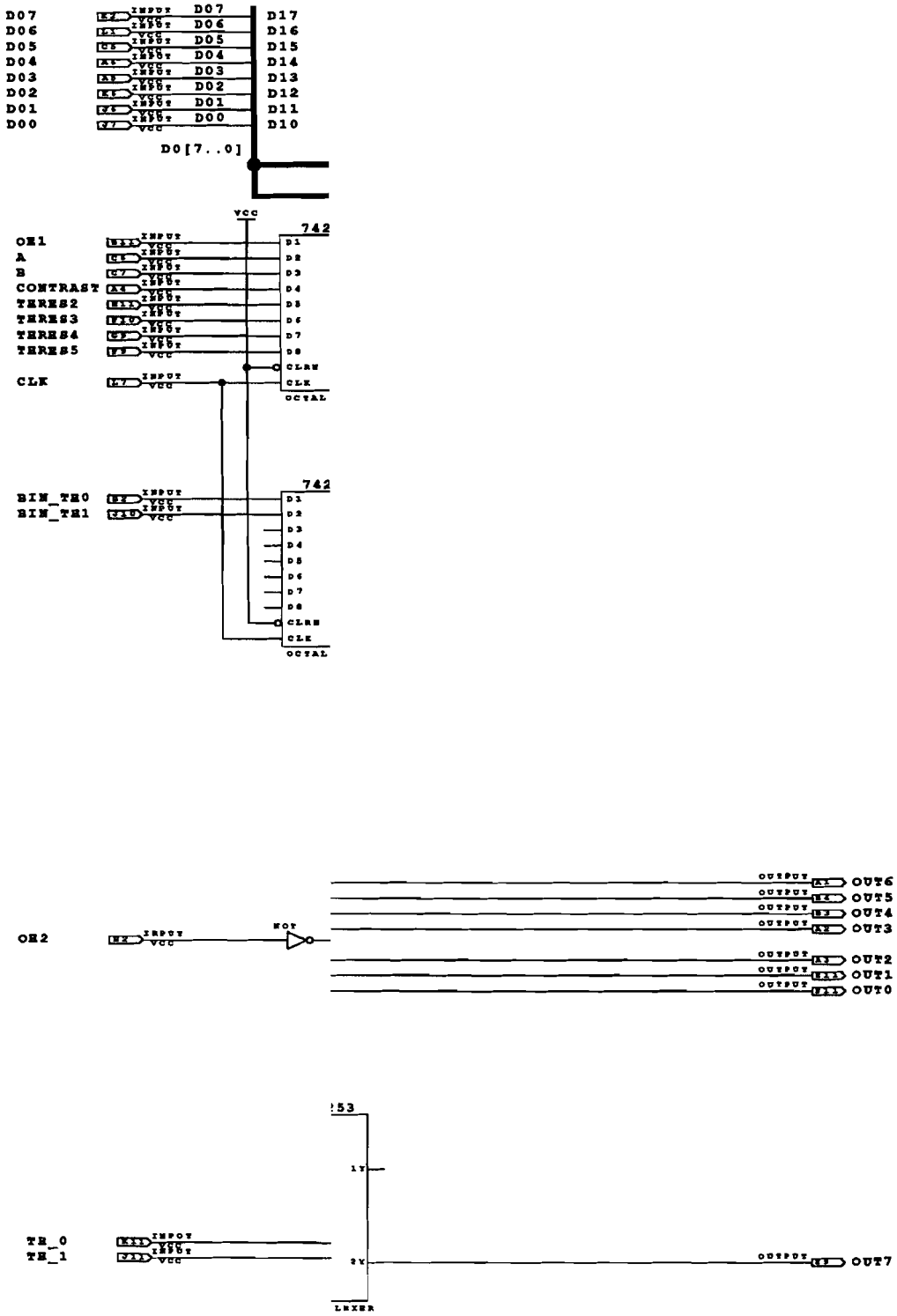

TMRN (PEC)
D5A
74LS14
D37
BIO 10W (PEC)
W0 (PEC)
MR (PEC)
74LS174
PEC
MAD[5..15] (PEC)
BIO [8..15] W (PEC)
CLK2
D17A
74F08
D17B
74F08
R4
47kE
R5
47kE
CLK3
DA[1..8] O (PEC)
D7
CLK1
SLIE (PEC)
16CUDSLR
EPM5032DC-2
D21
BIO 12 W (PEC)
BIO 13 W (PEC)
BIO [4..7] W (PEC)
BIO 3 W (PEC)
W5 (PEC)
BIO 15 W (PEC)
BIO 14 W (PEC)
SLIE (PEC)
BIO 11 W (PEC)
THRES[2..5]
CONTRAST
OUT[0..7]
DA[1..8] I
OUT_CONTR
PHILIPS CFT ISP IMI
Title
OPTIC TEST CIRCUIT
Size C
Document Number
JOERI VAN OOSTERHOUT
REV
Date: August 4, 1992
Sheet of



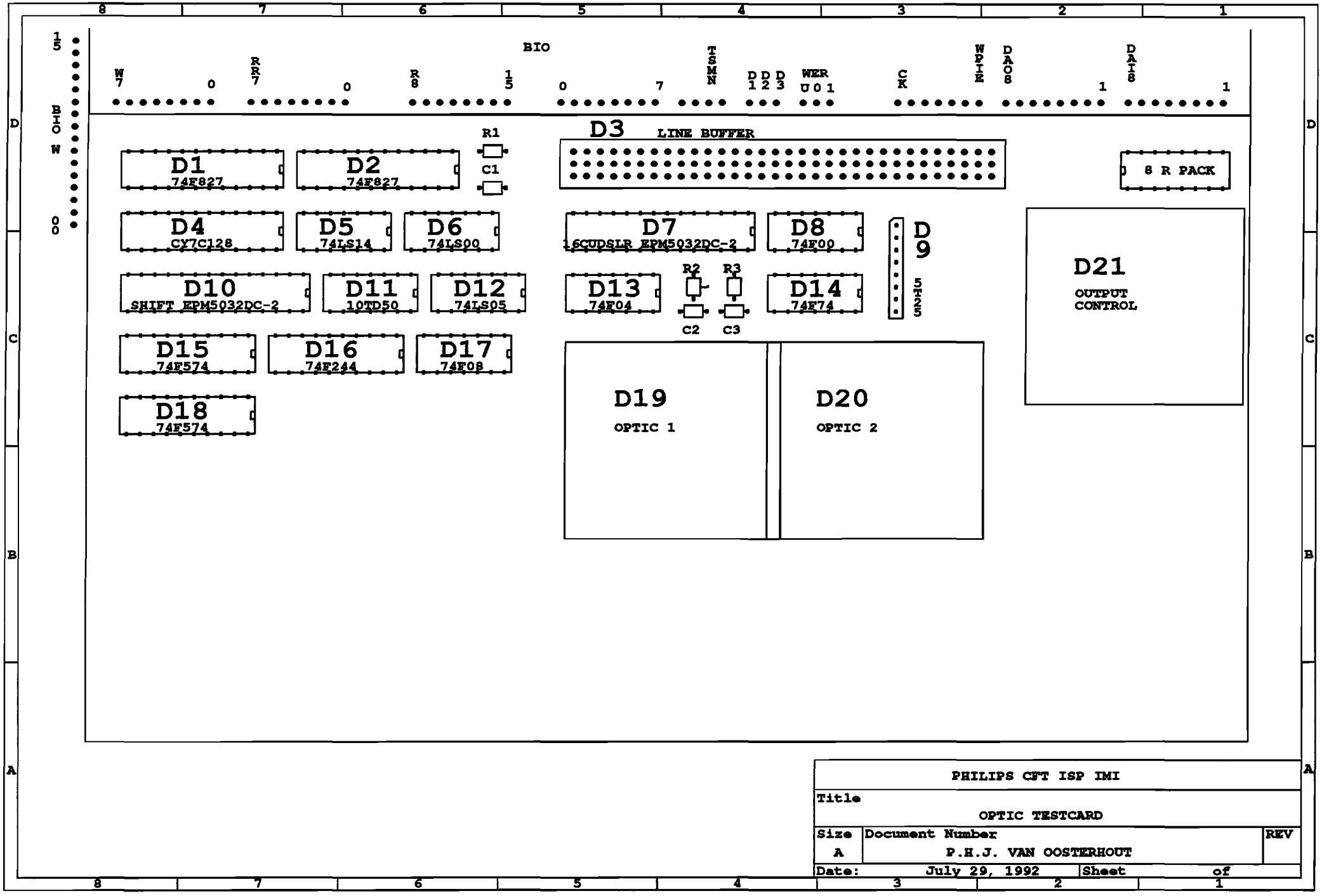


TITLE	OPTIC LOAD MACHINE		
COMPANY	PHILIPS - CFT		
DESIGNER	Joeri van Oosterhout		
SIZE D	EPLD EPM5032DC-1	NUMBER 1.00	REV A
DATE 2:33p 8-04-1992		SHEET 1 OF 1	
TURBO ON		SECURITY OFF	


VCC
SCLK (PEC)
D8A
&
74F00
R3
180E
C3
100pF
OPTIC
CLK1
CLK2
CLK3
D1
IN 5NS
10NS
15NS
20NS
25NS
5T25
R2
200E POT
C2
56pF
D13A
1
74F04
D8B
&
74F00
D14A
S
C1
1D
R
74F74
D13B
1
74F04
D13C
1
74F04
Size
A
Document Number
REV
Date: August 4, 1992
Sheet of



TITLE	OUTPUT CONTROL		
COMPANY	PHILIPS - CFT		
DESIGNER	P.H.J. van Oosterhout		
SIZE E	RPM5192GC-1	NUMBER 1.00	REV A
DATE 11:23a 7-31-1992		SHEET 1 OF 1	
TURBO ON		SECURITY OFF	



PHILIPS CFT ISP IMI		
Title: OPTIC TESTCARD		
Size: A	Document Number: P.H.J. VAN OOSTERHOUT	REV
Date: July 29, 1992	Sheet of	

N
M
L
K
J
H
G
F
E
D
C
B
A

PHILIPS CFT ISP EMI		
Title: OPTIC CONNECTION		
Size: C	Document Number: WIM REIJMANS	REV
Date: June 17, 1991	Sheet of	

APPENDIX B: Control Registers Testcard

Control REG 0

With the bits in this register it is possible to change the input of the card. Setting one bit in the register starts the handshake protocol between the card and the host when the Memory is written.

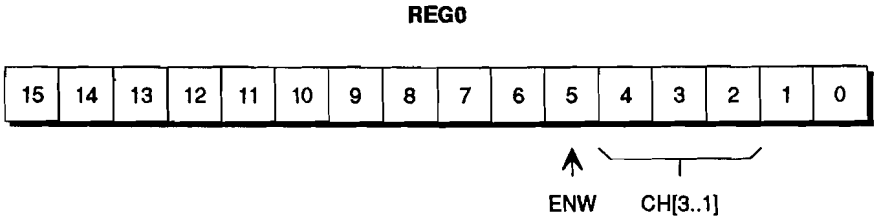


Figure D.1 *Bit configuration of REG 0*

Figure D.1 shows register 0 which controls the PEC decoder and table D.1 describes the bit configuration of register 0.

Table D.1 *Description of REG 0*

ENW	CH3	CH2	CH1	Description
x	1	0	0	Input from PI-3
x	0	1	0	Input from PI-2
x	0	0	1	Input from PI-1
0	x	x	x	Enable writing to Card
1	x	x	x	Disable writing to Card

Control REG 5

With the data in this register the outcome of the Output Controller can be influenced. The data also contains the threshold for making a binary output image.

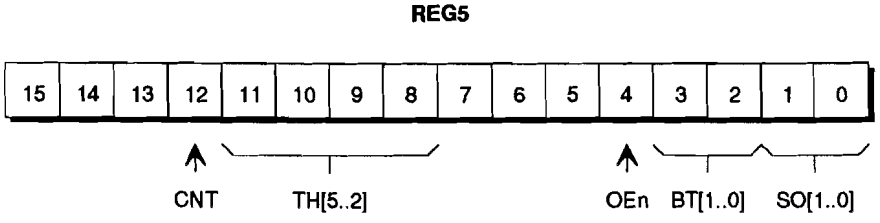


Figure D.2 *Bit configuration of REG 5*

Figure D.2 shows register 5 and table D.2 describes the bit configuration of the register.

Table D.2 *Description of REG 5*

CNT	TH[5..2]	OEn	BT[1..0]	S[1..0]	Description
0	x	x	x	x	No extra contrast
1	x	x	x	x	Extra contrast
x	Threshold	x	x	x	4 bit contrast threshold
x	x	0	x	x	Enable output Controller
x	x	1	x	x	Disable output Controller
x	x	x	0 0	x	Only bit 7 on PI-Bus
x	x	x	0 1	x	TH0 OPTIC 1 to PI-Bus
x	x	x	1 0	x	TH1 OPTIC 1 to PI-Bus
x	x	x	1 1	x	Not allowed
x	x	x	x	0 0	DO0 OPTIC 2 to PI-Bus
x	x	x	x	0 1	DO1 OPTIC 2 to PI-Bus
x	x	x	x	1 0	DO0 OPTIC 1 to PI-Bus
x	x	x	x	1 1	DO1 OPTIC 1 to PI-Bus

Control REG 6

The data in register 6 controls the loading of data in the two OPTICs. Setting one bit starts the parallel to serial conversion and the loading process. The register also contains the block address of the block that will be loaded in the OPTIC.

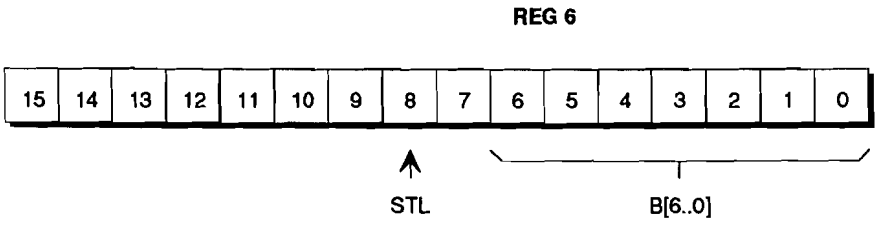


Figure D.3 *Bit configuration of REG 6*

In figure D.3 the bit configuration of the register is given. Table D.3 lists the effects of setting a certain bit of the register.

Table D.3 *Description of REG 6*

STL	B[6..0]	Description
0	x	Stop load OPTIC 1 & 2
1	x	Start load OPTIC 1 &2
x	Block address	Block number (0..127)

Control REG 7

Setting one bit in this register changes the OPTIC from WORK mode to LOAD mode. The data in register 7 also contains the CADDR. This address selects in LOAD mode the type of chain that will be loaded. In WORK mode CADDR selects one of the four processing windows.

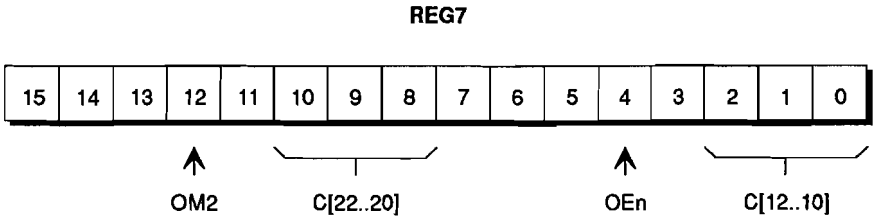


Figure D.4 *Bit configuration of REG 7*

Figure D.4 shows register 7 and table D.4 describes the bit configuration of the register.

Table D.4 *Description of REG 7*

OMx	C[x2..x0]	Description
0	x	WORK OPTIC x
1	x	LOAD OPTIC x
		Control address
x	0 0 0	OPTIC x :
x	0 0 1	window 0
x	0 1 0	window 1
x	0 1 1	window 2
x	1 0 0	window 3
		configuration

x = OPTIC number (1..2)