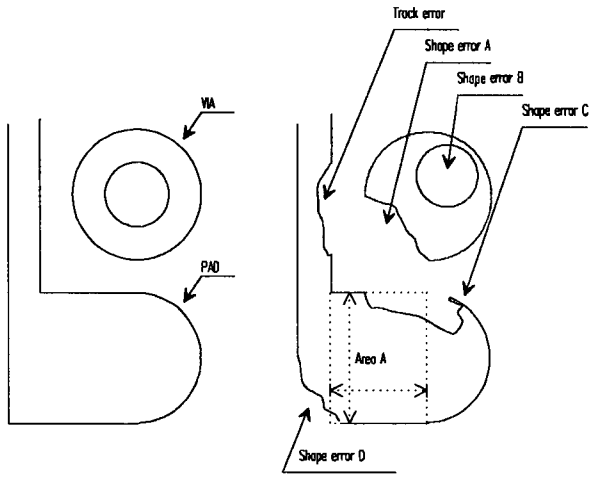
# Real-time model driven shape recognition

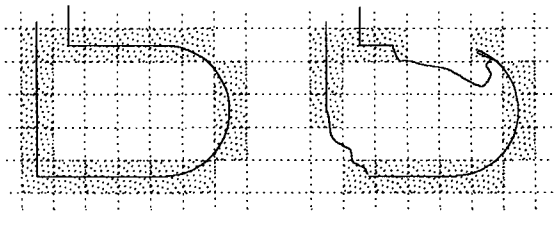
Heijmans, W.P.

*Award date:*
1991

# Real-Time Model driven Shape Recognition

by W.P. Heijmans (E 219015)



Eindhoven University of Technology
Department of Electrical Engineering
Digital Systems group (EB)

M.Sc. Thesis
Eindhoven, August 1991

commissioned and supervised by
Prof. ir. M.P.J. Stevens

supervion of
Ing. A.J.M. van Lier
Ir. A.C.M. Gieles

**Table of contents:**


Subject description . . . . . 6

Global approach . . . . . 8

Theoretical review of template matching methods . . . . . 10
- Introduction . . . . . 10
  - Static threshold . . . . . 10
  - Dynamic threshold . . . . . 10
- Binary template matching . . . . . 11
- Split-template binary template matching . . . . . 11
- Multi level binary correlation e.i. Dynamic Range Correlation . . . . . 12
- Multi-level input binary template matching using split-templates . . . . . 12
- Multi-level input binary template matching requiring a complete match . . . . . 13
- Non complete matches . . . . . 14

Test system setup . . . . . 16

Proposed CAD reference image generation algorithm . . . . . 18

First circuit syntheses . . . . . 19

32k byte line buffer . . . . . 20
- Functional description of the line buffer . . . . . 20
- READ MODIFY WRITE cycle . . . . . 21

Template Ram . . . . . 22

Override register . . . . . 24

CAD Control Logic . . . . . 25
- Loading the templates: . . . . . 27
- Activating the templates . . . . . 27
  - OPTIC . . . . . 27
  - IRIS . . . . . 27
  - CAD Control Input LUT . . . . . 27

Grid0..3 . . . . . 28
- Grid Offset . . . . . 28
- OPTIC Optimised Pixel Template Image Correlator . . . . . 29
  - Loading of the templates and configuration data . . . . . 34
- 2k Line Buffer . . . . . 36
  - Data reduction . . . . . 36
- Stored Template Ram . . . . . 40
- IRIS Image Recognition Integrated System . . . . . 41
  - Functional description . . . . . 42
  - IRIS modes . . . . . 42
  - Operation . . . . . 43
    - Binary Template Matcher Array . . . . . 43
  - Loading the 16*16 templates into the IRIS . . . . . 44
- Stored Template Ram for IRIS . . . . . 46

Output Logic . . . . . 47

Conclusions . . . . . 48

References . . . . . 49

Appendix . . . . . 51

**List of figures:**

Figure 1: ESPRIT project 2017 system architecture ..... 6
Figure 2: Principles of the Shape Feature Measurement Unit ..... 7
Figure 3: discrimination effect of two layered system ..... 8
Figure 4: one dimensional template matching with rulers ..... 9
Figure 5: Template variables ..... 10
Figure 6: Three land point ..... 10
Figure 7: Split black and white templates ..... 12
Figure 8: one dimensional edge ..... 13
Figure 9: DRC-module constructed from Rank Value Filters ..... 15
Figure 10: Paps system setup ..... 16
Figure 11: DRC module configuration ..... 17
Figure 12: simulation result of processing an image containing errors ..... 17
Figure 13: simulation result of processing an image containing no errors. ..... 17
Figure 14: Proposed algorithm ..... 18
Figure 15: magnification of shape centre ..... 18
Figure 16: Schematic overview of Shape Feature Measurement Unit ..... 19
Figure 17: principle line buffer circuitry ..... 20
Figure 18: line buffer timing diagram to enable *READ MODIFY WRITE* ..... 20
Figure 19: Template Ram memory map ..... 22
Figure 20: OPTIC template map ..... 22
Figure 21: block diagram for load mechanism ..... 23
Figure 22: timing of 1 shift cycle ..... 23
Figure 23: block diagram of override mechanism ..... 25
Figure 24: Input format description of Shape Feature Measurement Unit ..... 26
Figure 25: Grid enable signals ..... 28
Figure 26: block diagram of OPTIC architecture ..... 29
Figure 27: comparitor tree of sorting array ..... 31
Figure 28: sorting cell architecture ..... 32
Figure 29: cell 1 decision rules ..... 33
Figure 30: Construction of load chains for OPTIC processor ..... 34
Figure 31: template chain mapped on 8*8 window ..... 35
Figure 32: configuration chain in case of Multi-level Input Binary Template Matching ..... 35
Figure 33: block diagram of 2k Line Buffer ..... 36
Figure 34: recognition propagation ..... 36
Figure 35: normal operation timing of recognition capture ..... 37
Figure 36: recognition capture on grid (Address Change) ..... 37
Figure 37: hardware implementation of recognition capture ..... 38
Figure 38: timing of actual line buffer control ..... 39
Figure 39: physical block diagram of the IRIS ..... 41
Figure 40: single system block diagram ..... 42
Figure 41: system configuration of IRIS ..... 43
Figure 42: block diagram comparator cell ..... 44
Figure 43: usable IRIS configurations ..... 45
Figure 44: block diagram of Output Logic ..... 47
Figure 45: output LUT address format ..... 47

**List of tables:**

Table I: centre movement . . . . . 18
Table II: Destination codes for loading templates and configuration data of OPTIC's . . . . . 23
Table III: Window shapes . . . . . 29
Table IV: Possible pixel attributes . . . . . 30
Table V: supported operation modes . . . . . 33
Table VI: Template load chain addressing . . . . . 34
Table VII: Comparator cell logic output table . . . . . 44
Table VIII: map of template to IRIS memory . . . . . 45
Table IX: IRIS memory map . . . . . 45

**Subject description:**

This research concerns the development of a board for Shape Feature Measurement of so called "Via's" and "Pads" for multilayer Printed Circuit Boards. Via's are interconnection holes between the different inner layers and the pads are the connection points for the SMD components on the outer layers of the multilayer prints.

This board will be used in an image processing system for the inspection of "fine line" prints (track and isolation width approximately 40µm). This development is part of ESPRIT project #2017 concerning to Automated Process and Assembly Inspection by 3D Vision. Figure 1 give a schematic representation of the whole inspection system.

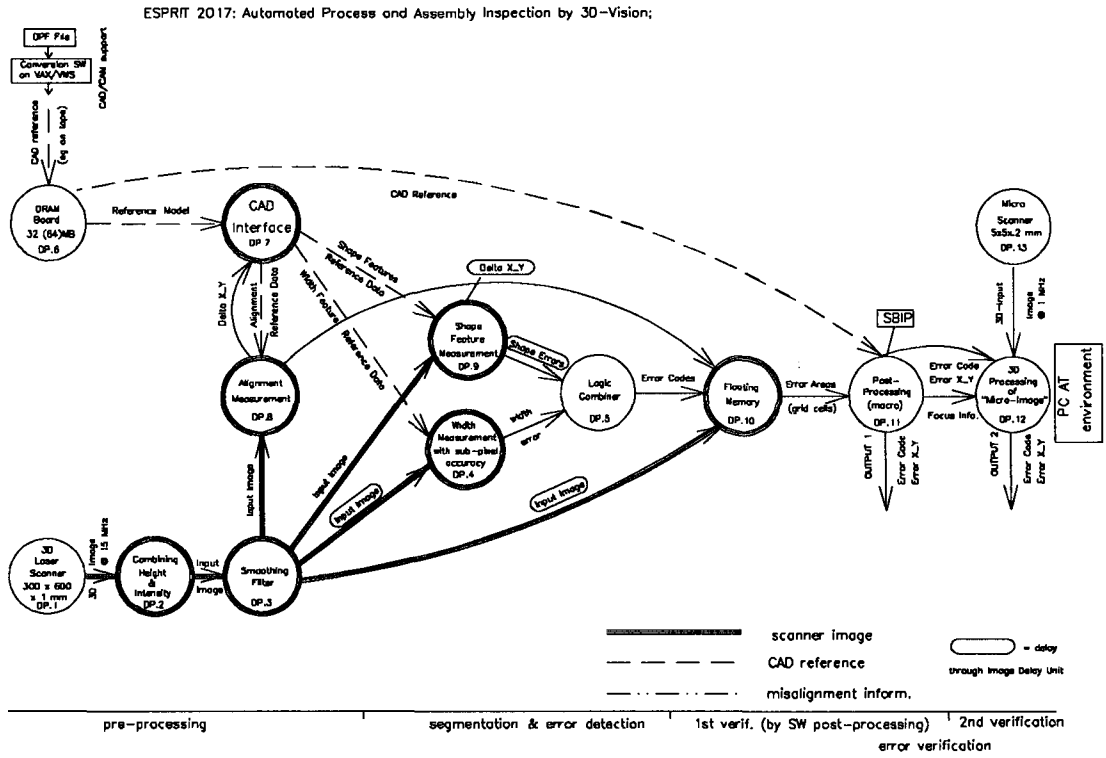



Figure 1: ESPRIT project 2017 system architecture

This board is required to run at a speed of 15 Mega pixels (1 pixel = 8 bits) per second (real time). The number of pixels to process is dependent on the size of the PCB panel under inspection, but can easily become several Giga pixels (board size 300mm * 600mm). The whole system will be CAD driven.

The shape feature recognition will be done with a layered system of two layers.

first layer : Shape recognition of elementary shape features like corners, arcs and other small features,with Pseudo Grey-level Correlation technic also called Dynamic Range Correlation.

second layer : The output of the first layer is a binary image, it serves as an input image to the second layer. In this image the elementary recognitions will be combined and processed with a Binary Correlator to see if the big shapes (pads and via's) are present and/or damaged.

Figure 2 show the two input images to the system. The fist image, CAD reference and the second image, scanner output image. In this example we damaged the pad and the via seriously to illustrate the kind of errors that can occur on these kind of shapes.

From the CAD reference image we can extract information about where to expect which features, so when we get the scanned image as it is drawn in Figure 2 the following errors will occur:

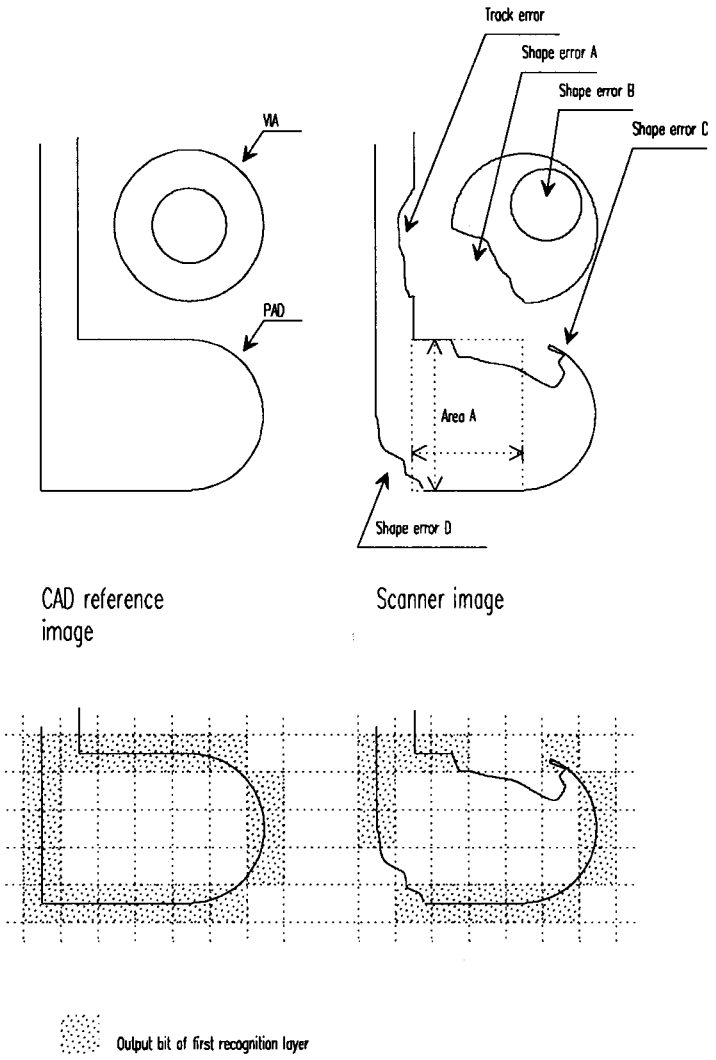



Figure 2: Principles of the Shape Feature Measurement Unit

| | |
|---|---|
| Track error: | This possible error should be recognised and reported by the Width Measurement Unit, which was specially made for measuring track and isolation widths. |
| Shape error A: | Part of the copper of the Via is missing. The Shape Feature Measurement Unit will report this as an error if too much copper is missing, according to the CAD data. |
| Shape error B: | The drill hole of the Via is out of centre. The Shape Feature Measurement Unit will produce an error if the remaining copper between the hole and the outer bound of the Via becomes to small according to the CAD data. |
| Shape error C, Shape error D: | Parts of the copper from the Pad are missing. These possible errors will be reported by the Shape Feature Measurement Unit if they are to extensive. However it is also possible that the area, indicated by "Area A", is scanned vertically by the Width Measurement Unit. |

The lower part of Figure 2 shows on the left side the CAD image and the generated second layer template, indicated by the hatched squares. On the right side one can see that parts of this pattern are missing so the second layer will not recognise this shape and will produce an error.

**Global approach**

From the subject description can be concluded that there is need for:

- Reference grid
- Filter for first layer
- Filter for second layer
- Interface between first and second layer
- Control logic for the filters
- Real-time approach

The need for a layered system is obvious since the currently available filters for multi level binary template recognition have only a maximum window size of 8*8 pixels. So for a shape size of 200*200 pixels it will impossible, without subsampling, to recognise it with one filter. A solution would be to create a matrix of filters and program the whole matrix with the template for the whole shape, producing one single bit to indicate the correctness of the shape. (But using 25*25 = 625 filters doesn't look good to start with.) Moreover it would not be possible to get a significant enough discrimination factor for missing features. A more realistic solution is just to use one 8*8 filter, reprogramming it with an elementary shapes, see if there is a recognition or not and collect the binary result in a memory for the second layer. So the first layer also performs in a way the grey level to binary conversion. As will be shown in the next chapter, using the Dynamic Range Correlation method will proof to be an elegant way of being insensitive to absolute grey values and as a result producing an intermediate binary image. The second layer will use this binary data as an input image, using binary correlators to make the decision if the total shape is correct or not. So this second layer will produce an extra data reduction, resulting in a single bit output for every shape. With this approach the ability of the circuitry to express the significance of the missing features will be much higher and gives also opportunity to eliminate the effect of occurring false edges. With a gradual change in the absolute grey values, static- and dynamic- threshold methods produce false edges, where as the DRC method only checks the range between MIN and MAX of a specified window.

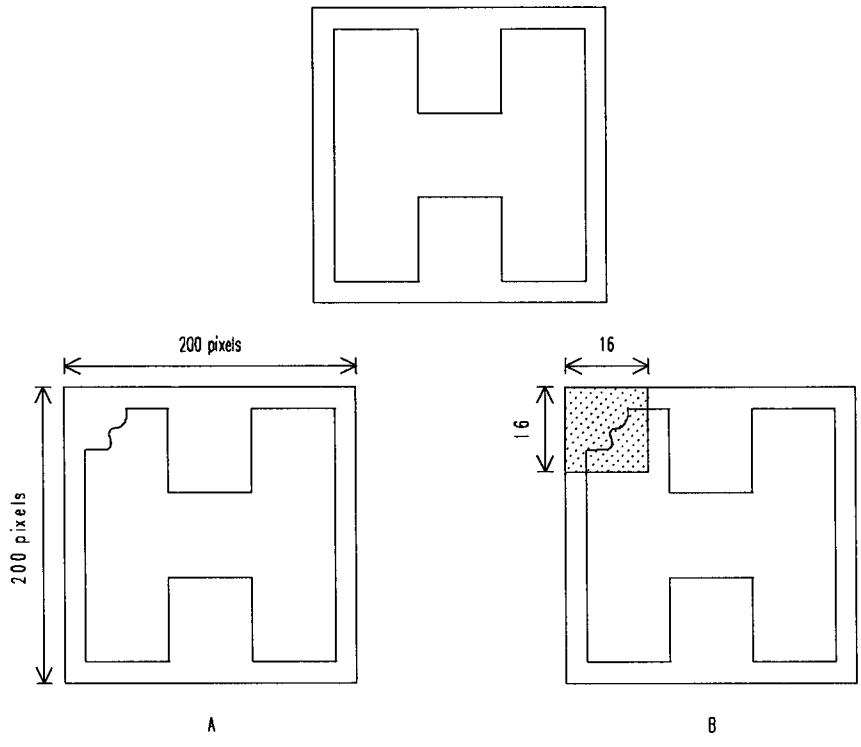



Figure 3: discrimination effect of two layered system

For example consider Figure 3. Suppose we a have filter of 200*200 pixels and a straight corner is missing, Figure 3 A, (e.g. 10 pixels), this is only 0.025% of the whole shape. The layered system, Figure 3 B, will produce on the same error, which is now (all 10 pixels in one 16*16 window) 256 pixels and checking only on 20 elementary features of the whole shape, gives 1/20 = 5% missing of the whole shape.

Because reprogramming of the filters takes some time, it is better to use more than one filter just to reduce the time caused by the reprogramming overhead.

We have chosen 2D template matching and not 1D because 1D template matching (rulers) have only a tolerance in object position perpendicular to the ruler as is shown in Figure 4. For a ruler in Y-direction the

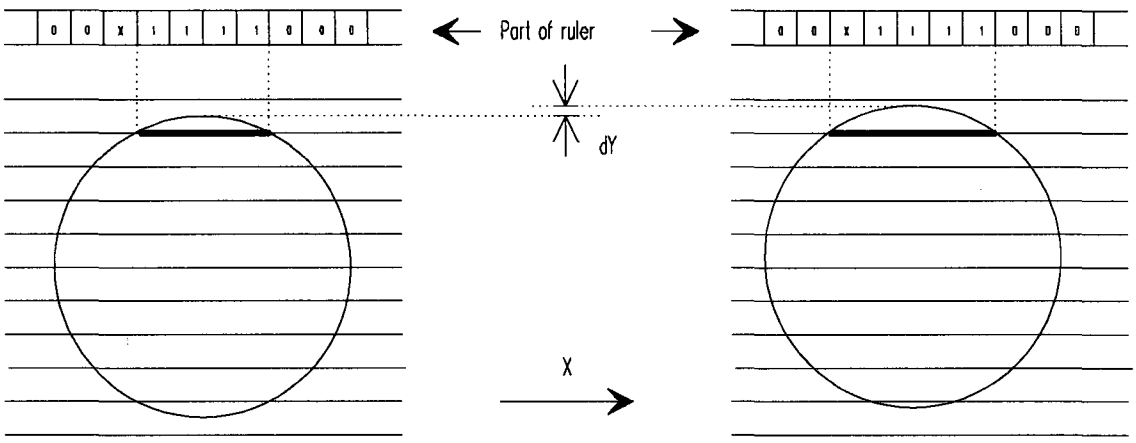



Figure 4: one dimensional template matching with rulers

same happens with a tolerance only in X-direction. Combining the two 1D rulers gives a 2D template which is able of handling a tolerance in object position in two directions at the same time.

Because of the necessity of a real time approach and in order to speed up the processing speed, we will make use of several banks of filters in parallel. If these banks are working on the same input image, only delayed in x and y directions, it will also be possible to recognise elementary features that are placed very close to each other (between 8 and 16 pixels depending on the x and y delay, both 8 in this case).

By placing a fixed grid over the image, of 16*16 pixels per cell, it will be possible to recognise features even if they are shifted maximum 4 pixels in all directions which is presumed to be enough because the input images will be aligned to each other. So if two features are less than 16 pixels apart we will have to use a different grid that is translated 8 pixels in x and/or y direction. This results in the use of four grids each of 16*16 pixel cells. Grid0(0,0), Grid1(8,0), Grid2(0,8) and Grid3(8,8).

First a review will be made on template matching methods to establish the ideas that are used in the different types of filters that will be used.

## Theoretical review of template matching methods

### Introduction

Static threshold

Template matching is a very powerful operation for object recognition in images. However its main disadvantage is that usually it requires a binary image as input. If the scenes to be processed are well-controlled and illuminated evenly, then it is usually possible to set a static video threshold, $T_{sv}$, to acquire a binary image, B(x,y), from a grey-value input image, G(x,y).

$$\begin{aligned} G(x,y) < T_{sv} &\rightarrow B(x,y) = 0 \\ G(x,y) \geq T_{sv} &\rightarrow B(x,y) = 1 \end{aligned} \tag{1}$$

Usually it is not possible to set a static video threshold in advance due to for example a non evenly distribution of the illumination or the reflectivity from objects may vary from scene to scene and from one place to an other in an image. For this reason it is not possible to find just one static video threshold which is valid for every position in the image.

Dynamic threshold

A possible alternative to static thresholding is to use dynamic thresholding. The algorithm for dynamic thresholding calculates the local video threshold, $T_{lv}(x,y)$, of a window, G(x,y), limited by -Left ≤ x ≤ Right and -Top ≤ y ≤ Bottom as follows:

$$T_{lv}(x,y) = \frac{1}{2} * \left( \underset{i=-Left}{\overset{Right}{MAX}} \underset{j=-Top}{\overset{Bottom}{MAX}} G(x+i,y+j) + \underset{i=-Left}{\overset{Right}{MIN}} \underset{j=-Top}{\overset{Bottom}{MIN}} G(x+i,y+j) \right) \tag{2}$$

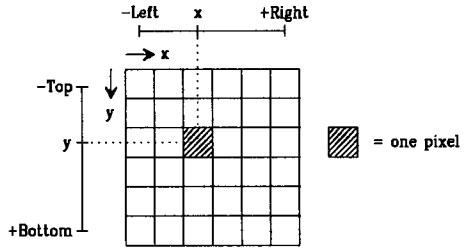



Figure 5. Template variables

Subtracting the thresholds from the input yields the following intermediate grey-value image $I_g(x,y)$:

$$I_g(x,y) = G(x,y) - T_{lv}(x,y) \tag{3}$$

Finally thresholding the intermediate grey-value image $I_g(x,y)$ at zero yields a binary output image B(x,y):

$$\begin{aligned} B(x,y) &= 1, \text{ if } I_g(x,y) > 0 \\ &= 0, \text{ if } I_g(x,y) \leq 0 \end{aligned} \tag{4}$$

However the binary images one gets using dynamic thresholding are not perfect. At so called "three-land points", the grey-level input image is too difficult to be represented in a single binary image. Figure 6. shows this situation.

At the positions A, B and C, it is easy to determine with dynamic thresholding which part of the outlined neighbourhoods is the locally darker part and which is the locally lighter part. When going along the left hand side to the border from A to C, we start and end in a locally "light" neighbourhood. We start and end in a locally "dark" neighbourhood when going from B to C along the bottom right hand side of the border. But when going from A to B, we start in a locally "dark" area and end in a locally "light" area. So in one way or another, an extra "false" zero-crossing between A and B has to be produced. This is indicated with the dotted line in Figure 6..

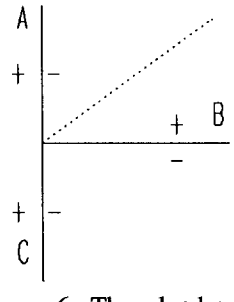



Figure 6.: Three land point

When a static threshold is applied to get the binary image, this problem does not occur. So another alternative is to threshold the input image with every static video threshold possible (256 in case of a 8 bit input image) and to process each of these images with binary correlation to find the features we are looking for. Although this method does not have the above mentioned problems it either takes a lot of time if this operation is preformed serially per video threshold, or takes a lot of hardware if all the possible binary images are processed in parallel.

A satisfiable solution has been found in the so-called DRC-method (Dynamic Range Correlation) also referred to as Multi Level Binary Template Matching.

**Binary template matching**

In binary template matching, a template is correlated with an input image. So a template is put on an input image and shifted from pixel to pixel. For every position that the template matches we have a recognition and the output is a binary 1. If there is no match the output is a binary 0. Every template consists of two matrices with binary elements, namely the "white"/"black" matrix T(x,y) and the "do care"/"don't care" matrix D(x,y) which are filled as stated in equations (6) and (5)

$$\begin{aligned} D(x,y) &= 1; \textit{ template element x,y is do care} \\ &= 0; \textit{ template element x,y is don't care} \end{aligned} \tag{5}$$

$$\begin{aligned} T(x,y) &= 1; \textit{ template element x,y is white, if } D(x,y)-1 \\ &= 0; \textit{ template element x,y is black, if } D(x,y)-1 \end{aligned} \tag{6}$$

The matrices are limited in size : -Left ≤ x ≤ Right; -Top ≤ y ≤ Bottom.

$$O_g(x,y) = \sum_{i=-Left}^{+Right} \sum_{j=-Top}^{+Bottom} \overline{I(x+i,y+j) \oplus T(i,j)} \bullet D(i,j) \tag{7}$$

so $O_g(x,y)$ is limited by:

$$0 \le O_g(x,y) \le \sum_{i=-Left}^{+Right} \sum_{j=-Top}^{+Bottom} D(i,j) - S \tag{8}$$

Where S is the number of do care elements of the template.

We can define a recognition or a match to be at the positions where $O_g(x,y)$ equals S, which implies complete equality of input and template. But in order to be able to cope with a little bit of noise and deformations, we define that we have a recognition where $O_g(x,y)$ is higher than or equal to a correlation threshold $T_c$. So the binary output image, B(x,y), of binary template matching is given by equation (8)

$$\begin{aligned} O_b(x,y) &= 1; \textit{ if } O_g(x,y) \ge T_c \\ &= 0; \textit{ if } O_g(x,y) < T_c \end{aligned} \tag{9}$$

and $1 \le T_c \le S$, to have a meaningful Threshold.

We will use the IRIS (Image Recognition Integrated System) from S3 (Silicon & Software Systems) configurated as 2 * (2* (16*16)) binary template matcher. Technical data can be found in the appendix.

**Split-template binary template matching**

In the preceding section, we used one template which contained black do care elements as well as white do care elements. In some cases it is better to split the template into two, one containing only black do care

elements and don't cares, and the other template only white do care elements and don't cares. Both sub-templates have to match on the same position in order that the complete template matches. The advantage of this approach is that it is much easier now to handle situations where the number of white and black do care elements differ a lot. Consider for example the template of a thin line in the next figure.

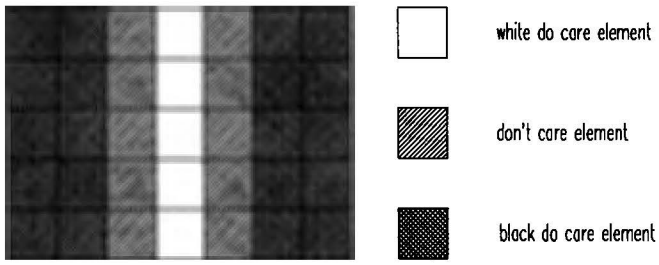



Figure 7.: Split black and white templates

A complete match requires that all of the 25 do care elements match with corresponding pixels of the input image. However, if we want to tolerate some noise, then we can lower the correlation threshold. If we allow for 5 noise pixels, then this template is also recognised on a constant black input. This is clearly not desirable. Making the correlation threshold one higher does not help to much. Now, isolated white pixels are also recognised as a white line. Splitting the template into a black part and a white part with separate threshold as well gives us much more control. If the threshold for the white sub-template is set such to allow one noise pixel and the threshold for the black sub-template for 4 noise pixels, then we allow again for 5 noise pixels in total, but constant black areas or isolated white pixels will not be recognised as vertical white lines any more.

**Multi level binary correlation e.i. Dynamic Range Correlation**

As stated before, it is sometimes very difficult to find a single static video threshold in order to get a good bi nary image in which one can recognise objects using binary template matching. A good solution to this problem is to threshold the grey-level image with every static video threshold possible. Each of the resulting binary images is then used as input for binary template matching. The recognition in each of these binary images can be added pixel-wise, to get the final output. The value of the final output on a certain position is now dependent of the contrast of object and background. The final output can be thresholded with a threshold in order to separate high contrast objects from low contrast ones.
In the next section will be shown that "split-template binary template matching" on all possible static video thresholds on the grey-level input image can be done simultaneously. First we will look at the behaviour of the matching of the sub-templates when the video threshold $T_v$ is varied.

**Multi-level input binary template matching using split-templates**

When we are using a split-template, we have two sub-templates, the black sub-template, $T_b(x,y)$, with correlation threshold $T_{cb}$ and a white sub-template, $T_w(x,y)$ with correlation threshold $T_{cw}$. With $T_{cb}$ and $T_{cw}$ between 1 and the value needed for complete match. If we take the black sub-template and apply every possible video threshold we will get the following effect:

$$\begin{aligned} I_b(x,y) &= 1 \ (\textit{white}) \ ; \ \textit{if} \ G(x,y) \geq T_v \\ &= 0 \ (\textit{black}) \ ; \ \textit{if} \ G(x,y) < T_v \end{aligned} \qquad (10)$$

Suppose we apply the highest video threshold $T_v$. All of the resulting binary image will become black and any black sub-template will match everywhere in this binary image (all 0). Lowering $T_v$ step by step will result in binary images that will contain more and more white pixels, but never will an already white pixel become black when lowering $T_v$. Lowering $T_v$ to the lowest value (0) will yield a binary image which is completely white. So no black sub-template will fit anywhere in this image. This means that for every

location (x,y) there is a video threshold $T_{vb}(x,y)$ resulting in a binary image for which the black sub-template will still match at location (x,y), but not any more for the binary image resulting from $T_{vb}(x,y)-1$. This means that the black sub-template matches on location (x,y) in all the binary images resulting form the application of video thresholds in the interval $[T_{vb}(x,y),max]$, but it will not match on that location in all of the binary images resulting from the application of all video thresholds in the interval $[0,T_{vb}(x,y)-1]$.

In the same way it is possible to show that the white sub-template will match on location (x,y) in all the binary images resulting from the applications of video thresholds in the interval $[0,T_{vw}(x,y)]$ but not in the binary images resulting from the application of video thresholds in the interval $[T_{vw}(x,y)+1,max]$. These intervals are depicted in the next figure for an one dimensional signal.

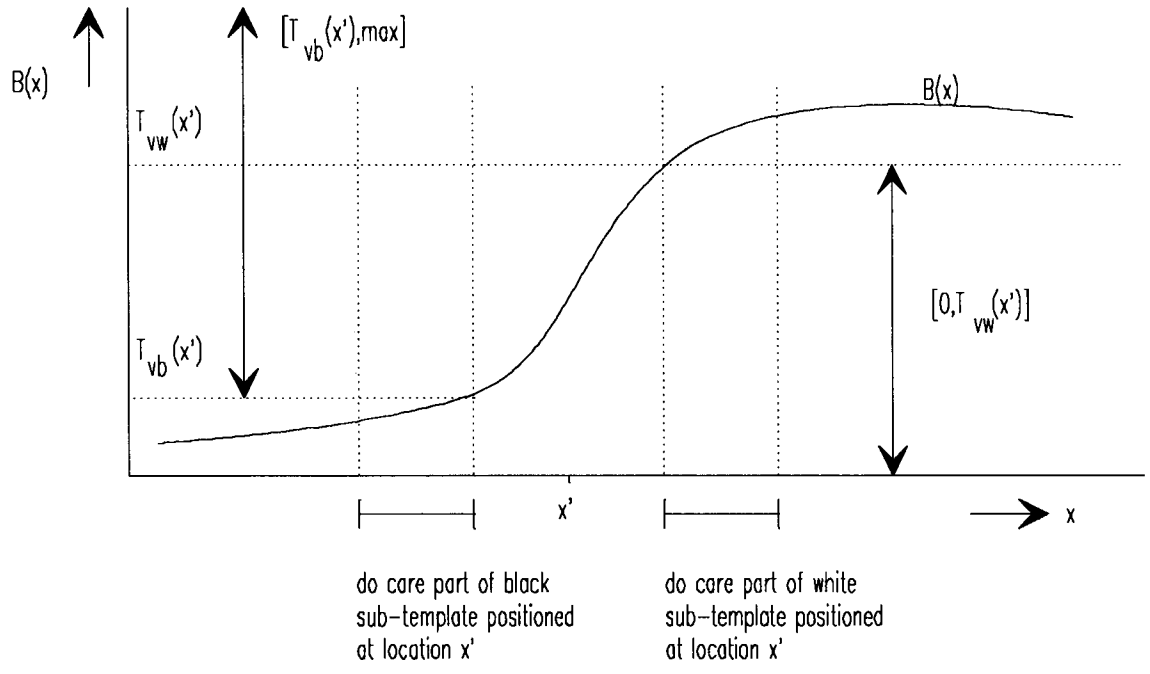



Figure 8.: one dimensional edge

If there is an overlap between the intervals $[T_{vb}(x,y),max]$ and $[0,T_{vw}(x,y)]$, then the complete template will match in all of the binary images resulting from the application of video thresholds in the interval $[T_{vb}(x,y),T_{vw}(x,y)]$. The complete template will therefore match on $T_{vw}(x,y)-T_{vb}(x,y)+1$ video levels. This number is called the range. So all that has to be done is finding $T_{vw}(x,y)$ and $T_{vb}(x,y)$ for every location (x,y), compute the range and checking whether this range is greater than or equal to one or some other non-negative range threshold.

**Multi-level input binary template matching requiring a complete match**

Again we use a split-template with $T_b(x,y)$ with correlation threshold $T_{cb}$ and a white sub-template $T_w(x,y)$ with correlation threshold $T_{cw}$.

Suppose furthermore that the correlation thresholds $T_{cw}$ and $T_{cb}$ are set such that a complete match is required for both. Looking at the former figure gives the strong feeling that the maximum over the input pixels corresponding to the do care elements of the black sub-template positioned at location (x,y) has something to do with $T_{vb}(x,y)$.

Applying a video threshold which is one higher than the maximum of a set of pixels results in a binary image that is black for all of these pixels, when we use video thresholding as defined in equation (10). If this set of pixels corresponds with the set of the do care pixels of the black sub-template positioned at location (x,y), then we have a complete match using this video threshold. If we lower the video threshold with one,

then at least the pixel which has the maximum value will become white in the resulting binary image. This means that we do not have a complete match any more and that therefore the maximum of the input pixels corresponding with the do care pixels of the black sub-template is just one lower than the $T_{vb}(x,y)$ we are looking for. In an equation:

$$T_{vb}(x,y) = 1 + \underset{\forall(i,j):T_b(i,j)=1}{MAX} G(x+i,y+j) \tag{11}$$

Likewise, applying a video threshold which is one higher than the minimum of a set of pixels results in a binary image that is white for all of these pixels. If this total set of pixels corresponds with the set of the do care pixels of the white sub-template positioned at location (x,y), then we have not a complete match using this video threshold. If we lower the video threshold with just one, then the pixels having the minimum value will be thresholded to white as well and now we have a complete match. Therefor, the minimum of the input pixels corresponding with the do care pixels of the white sub-template is the $T_{vw}(x,y)$ we are looking for:

$$T_{vw}(x,y) = \underset{\forall(i,j):T_w(i,j)=1}{MIN} G(x+i,y+j) \tag{12}$$

Now we have shown that for templates requiring complete matches, we can find $T_{vw}(x,y)$ and $T_{vb}(x,y)$ by a filter that computes the minimum and the maximum value of a programmable set of pixels in a window. The range or the number of video thresholds for which both sub-templates match equals:

$$\begin{aligned} N_{T_v}(x,y) &= T_{vw}(x,y) - T_{vb}(x,y) + 1 \\ &= \underset{\forall(i,j):T_w(i,j)=1}{MIN} G(x+i,y+j) - \underset{\forall(i,j):T_b(i,j)=1}{MAX} G(x+i,y+j) \end{aligned} \tag{13}$$

In order to have a match of the total template, $N_{T_v}$ should be greater then zero.

**Non complete matches**

In this section we will relax the constraints from the preceding section on the correlation thresholds. We will allow $m_b$ mismatches for the black sub-template and $m_w$ for the white sub-template. Setting both $m_b$ and $m_w$ to zero gives the situation of the preceding section.

First we define the value "maxbut n" of a set of numbers. We first sort the set in descending order. The value "maxbut n" of a set of numbers is the value of the $n^{th}$ element of the ordered set, when counting starts at zero. So "maxbut 0" is the maximum value of a set of numbers. Similarly, "minbut n" is defined, but then the set is sorted in ascending order.

Suppose, we apply a video threshold which is one higher than the "maxbut $m_b$" of a set of pixels. This means that there can be from zero to $m_b$ pixels with a value higher than or equal to this threshold, but certainly not more than $m_b$. The binary image that results from this video threshold might have been thresholded to white for at most $m_b$ highest valued pixels, but is will certainly be black for the rest of the pixels of the set. If this set corresponds with the do care pixels of the black sub-template, positioned at location (x,y), then we have at most $m_b$ mismatching elements, which is allowed, so we have a match. If we lower the video threshold by one, then the "maxbut $m_b$" of the set of pixels will be equal to the new video threshold, which means that there are now at least $m_b+1$ pixels higher than or equal to the video threshold. These will all be thresholded to white which means that we have at least one to many mismatches. Therefore, the "maxbut mb" of the input pixels corresponding with the do care pixels of the black sub-template is just one lower than the $T_{vb}(x,y)$ we are looking for when we allow $m_b$ mismatches.

$$T_{vb}(x,y) = 1 + RANK\{(N(R_b(x,y)) - 1) - m_b \,,\, R_b(x,y)\} \tag{14}$$

with:

| | |
|---|---|
| $R_b(x,y)$ | the set consisting of the pixels G(x+i,y+j) for which $T_b(i,j)=1$, |
| N{S} | the number of elements of the set S and |
| RANK(n,S) | the $n^{th}$ element of the set S ordered from lowest to highest value; counting starts at 0, |

therefore: RANK(0,S) = MIN S and RANK(N{S}-1,S) = MAX S

Likewise, the "minbut $m_w$" of the input pixels corresponding with the do care pixels of the white sub-template when we allow $m_w$ mismatches, will give us a value which is $T_{vw}(x,y)$. In formula:

$$T_{vw}(x,y) = RANK\{m_w, R_w(x,y)\} \tag{15}$$

A hardware implementation of a DRC-module is initially made with 2 RVF's (Rank Value Filters) L64220 from LSI-Logic. The configuration as it is shown in Figure 9. was used because the OPTIC (Optimized Pixel Template Image Correlator which is a pipelined Max-Min Filter image processor) containing a 8*8 multi level binary template matcher with 4 addressable templates, will not be available until the end of june 1991.

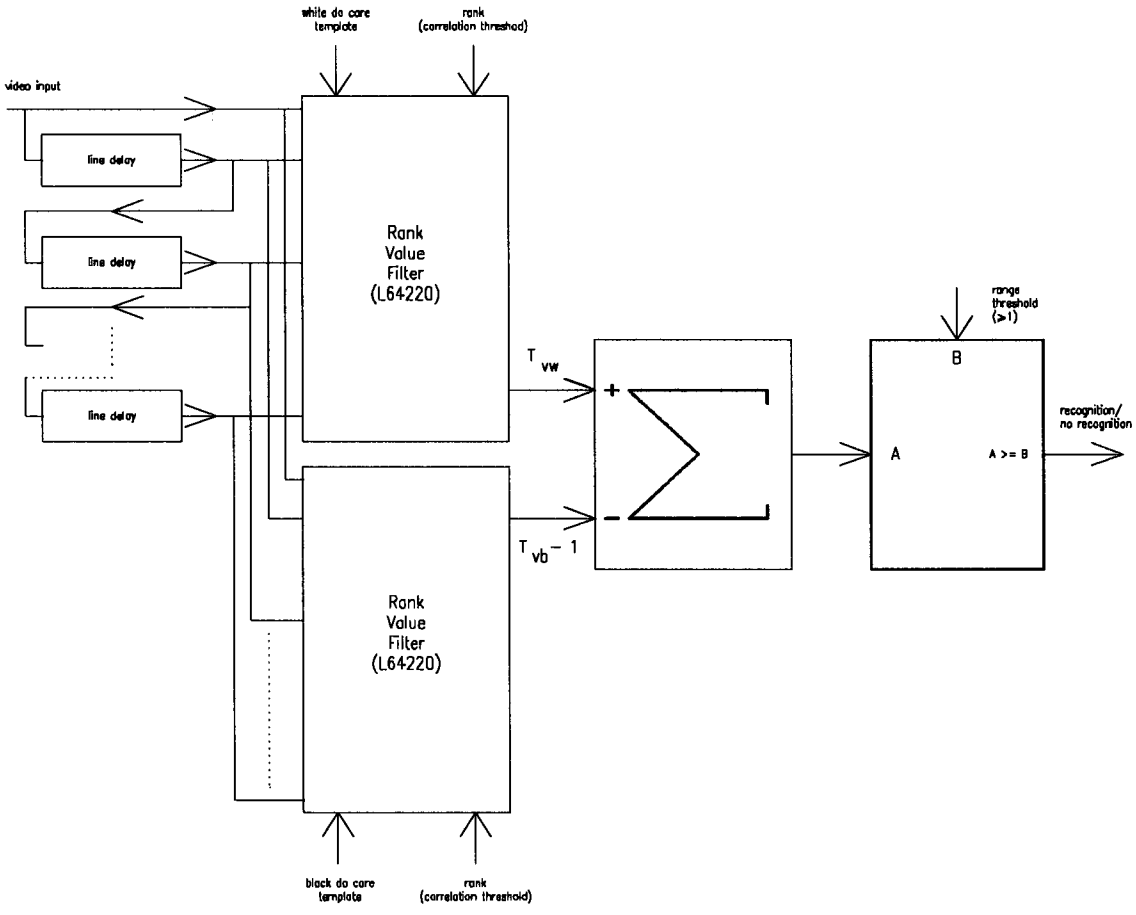



Figure 9.: DRC-module constructed from Rank Value Filters

The actual Shape Feature Measurement Unit will be build with 8 OPTIC's, two for each grid. For more specific information see the appendix.

The next chapters will handle the hardware design of the board.

**Test system setup**

PAPS: Picture Aquisition and Processing System, Philips inhouse development for high-end machine vision applications.

Figure 10. shows the PAPS system configuration as it is used for testing purposes leading to the synthesis of circuitry parts for the Shape Feature Measurement Unit.

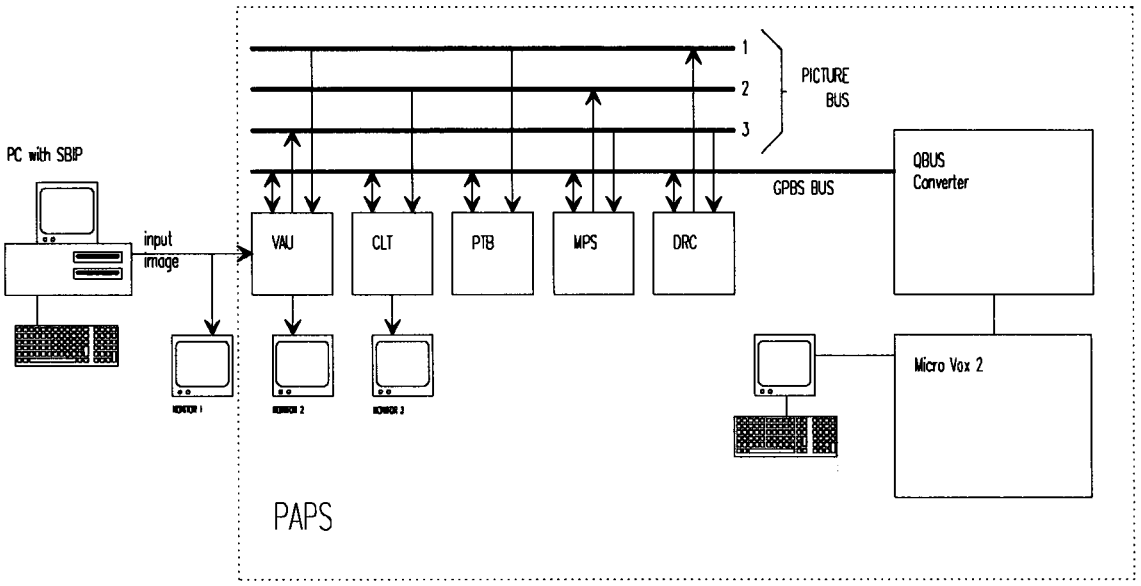



Figure 10.: Paps system setup

The PC is used to generate input images for the PAPS system. These images can be artificial or real (stored on disk). Monitor 1 shows the input image.
The PAPS system is configured with the following hardware:
VAU : Video Acquisition Unit that samples the incoming image and generates the appropriate timing signals on the control bus.
CLT : Color Look up Table for displaying a grey-value image in pseudo colors.
PTB : Picture Transfer Buffer for transferring picture data to the host computer, the MicroVax 2.
MPS : Mapped Picture Store to store intermediate image related results.
DRC : Dynamic Range Correlator (main function). This board contains the hardware parts which are needed to simulate the functions to be implemented.

The DRC board, Figure 11., contains two Rank Value filters, for the Dynamic Range Correlation, a data reduction line buffer and an IRIS. For a detailed circuitry layout see the appendix.

Since we only have one DRC-module available it is not possible to test the whole function of the Shape Feature Measurement Unit real time. As a result of this a program is made (CAD2.TPL) to simulate the Shape Feature Measurement Unit. As input for the simulation a static image is used.
The testprogram does the following:

1. program DRC with a sub shape (template)
2. place window on grid cell position
3. gather recognition points and if recognition takes place, store a point in the MPS on the corresponding grid position
4. repeat step 1. to 3. for all sub shapes of a bigger shape, until all data is checked
5. get information from MPS and see if whole shape is recognised

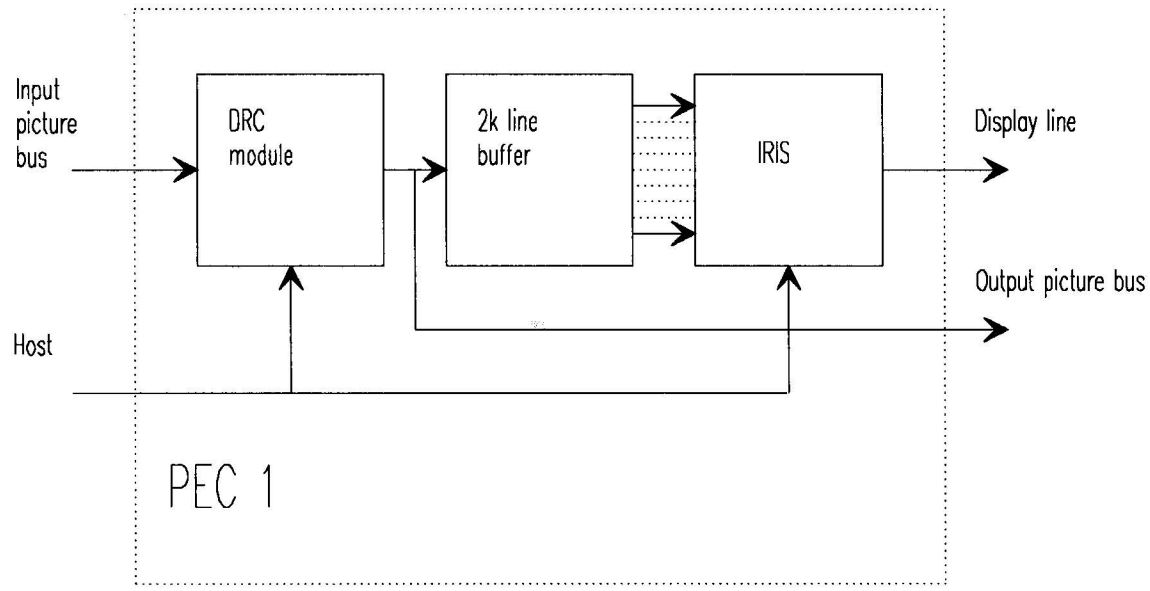



Figure 11.: DRC module configuration

It is not possible to see the results of the second layer recognition (IRIS in Figure 11.) at this time because the elementary templates are applied in sequence and not in parallel, so the relevant data is not available at the right moment. But after storing the recognition points in the MPS, the MPS image could be used to send to the IRIS.

The result image as it appears on monitor 3 of Figure 10. looks like Figure 12. and Figure 13.

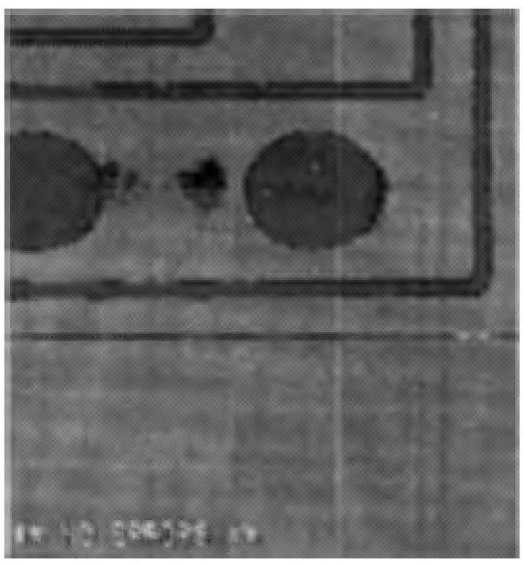


Figure 13.: simulation result of processing an image containing no errors.

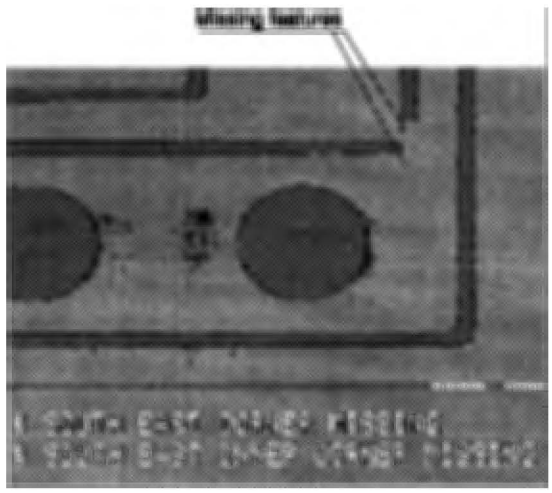


Figure 12.: simulation result of processing an image containing errors

Figure 12. shows the situation with a missing piece of copper on the corner of the track, and Figure 13. shows the same situation without errors.

**Proposed CAD reference image generation algorithm**

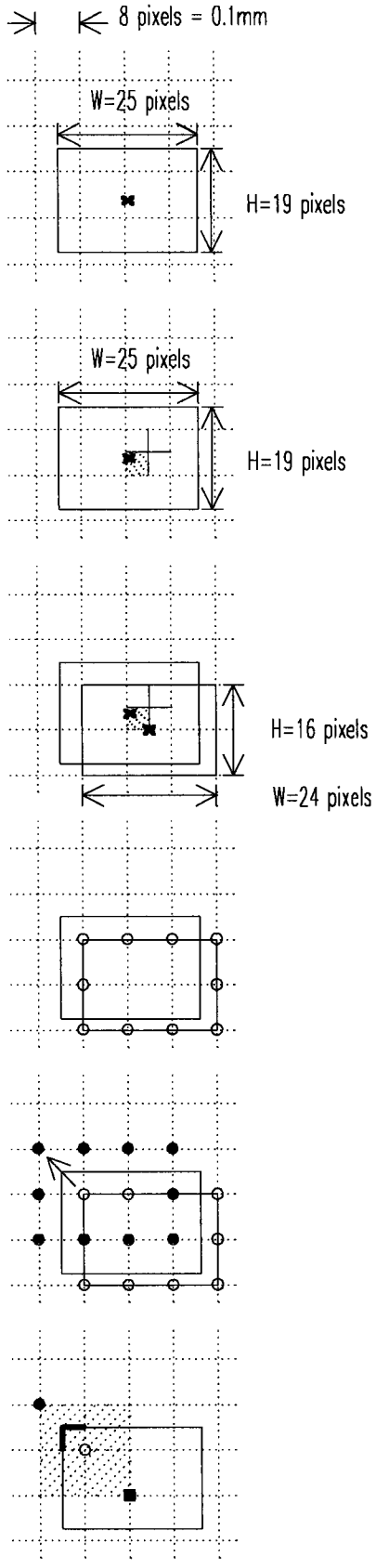



Figure 14.: Proposed algorithm

Figure 14. shows the proposed algorithm to generate the CAD reference image.

Original shape:

Determine the Width and Height and (X) centre position of the shape.

Determine which subgrid cell contains the centre of the shape and in which quadrant of the subgrid cell the centre is located. Round the Width and Height to the nearest subgrid value (multiple of 8 pixels). E.g. 19 will become 16 and 31 will become 32 (this is true for horizontal and vertical rounding).

Depending on the calculation (Width DIV 8) the result will be odd or even likewise for (Height DIV 8), resulting in following table.

**Table I**: centre movement

| W / 8 | H / 8 | Centre to |
|---|---|---|
| odd | odd | B |
| odd | even | D |
| even | odd | A |
| even | even | C |

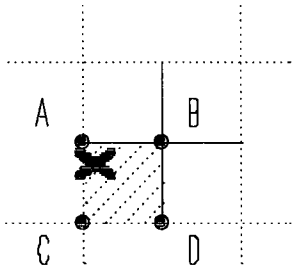



Figure 15.: magnification of shape centre

Create reference shape around new centre by placing rectangles with width and height multiples of 8 pixels on the reference shape. The outer corners of these rectangles mark the template positions of the reference shape.

Translation of the template mark points with -1,-1 gives the template enable positions on the subgrids. By assigning these enable points to the appropriate grid and labelling them with the intended template code results in the CAD reference input image for the first layer of the Shape Feature Measurement Unit.

So one enable point will enable a certain template for 16*16 pixels, as indicated in the figure by the hatched area. The black square in the right bottom corner of the hatched area, marks the read out position of the recognition. If a recognition took place there will be a 1 if not there will be a 0.

## First circuit syntheses

Concluding from the previous chapters and specifications we come to the following schematic layout for the Shape Feature Measurement Unit (Figure 16.).

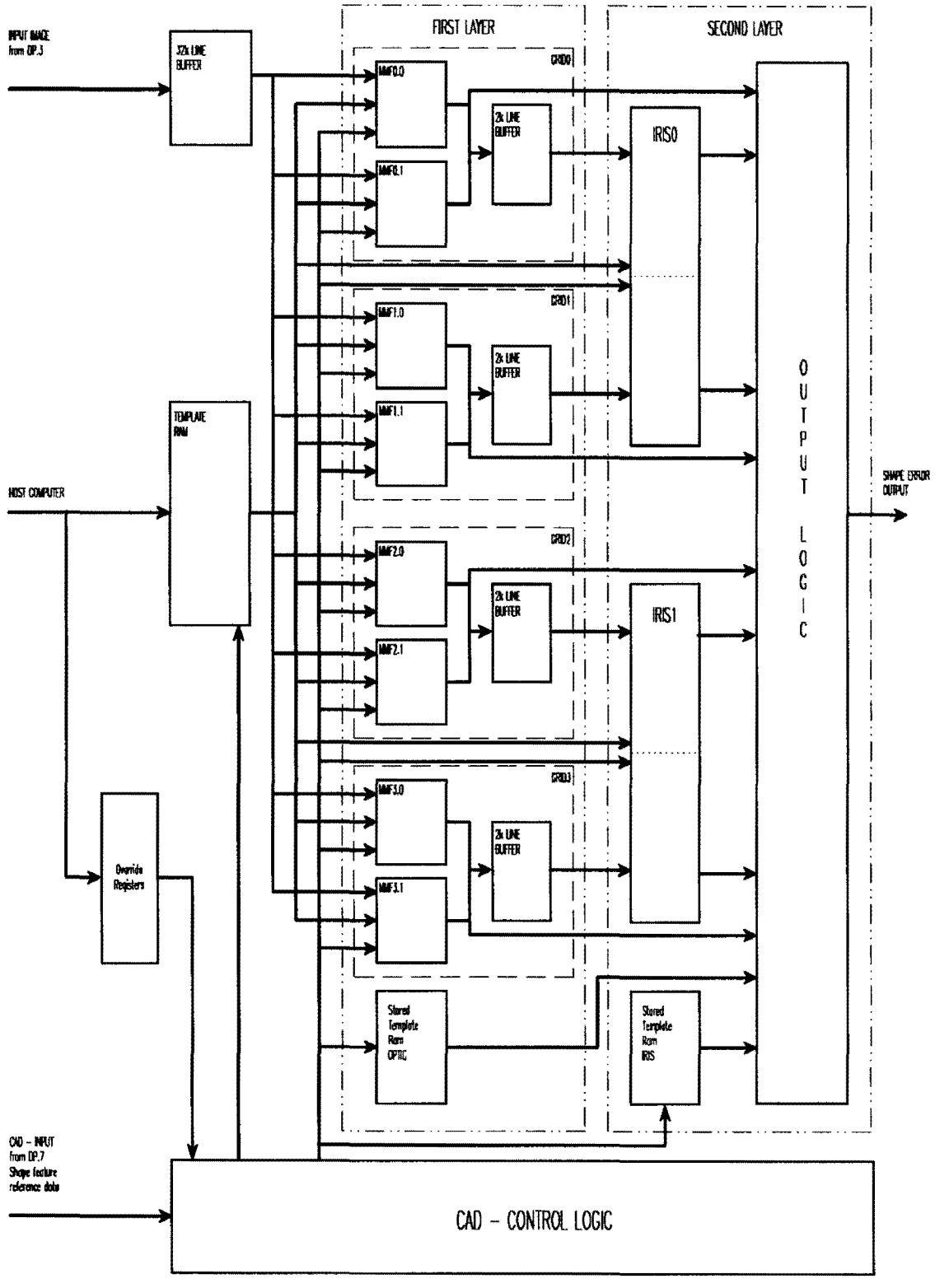



Figure 16.: Schematic overview of Shape Feature Measurement Unit

### 32k byte line buffer

The input signal from the laser scanner is buffered by a 8 lines line buffer of 32k pixel per line. The purpose of this buffer is to be able to supply the first layer filters (OPTIC's) with an input image. The PAPS picture data is an 8 bits wide value, this data is appearing line wise on the picture bus. Since the OPTIC's need 8 bytes of input data every clock cycle it is necessary to have these 8 byte of data ready at every clock cycle. A hardware mechanism that satisfies this requirement is shown in Figure 17.

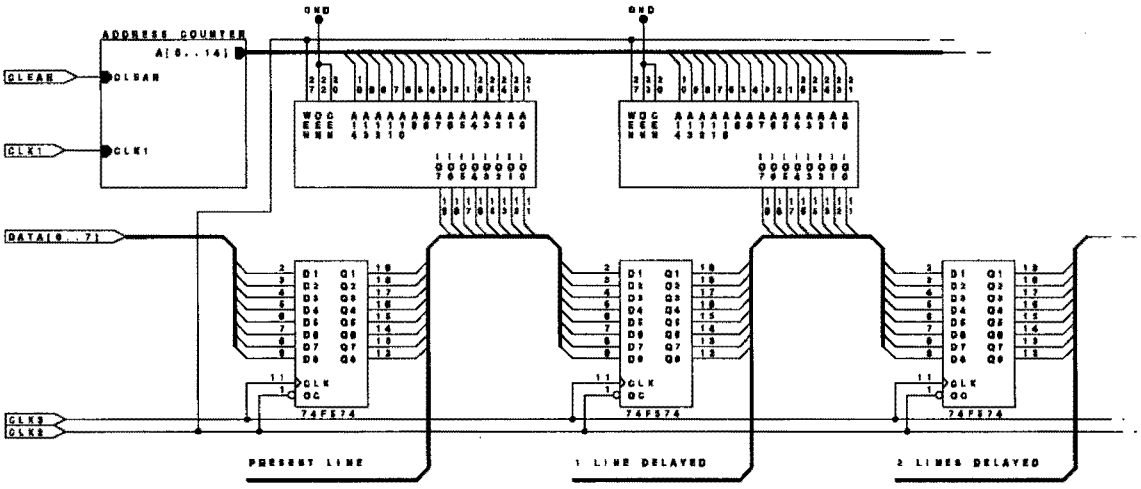



**Figure 17**: principle line buffer circuitry

Functional description of the line buffer.

As can be seen in Figure 17 the line buffer consists of a memory and a latch for each delayed line, an address generator and one input latch. The input signals: CLK1, CLK2 and CLK3 are clock signals that enable *READ MODIFY WRITE* actions to be taken on the memories; CLEAR is a reset signal for the address

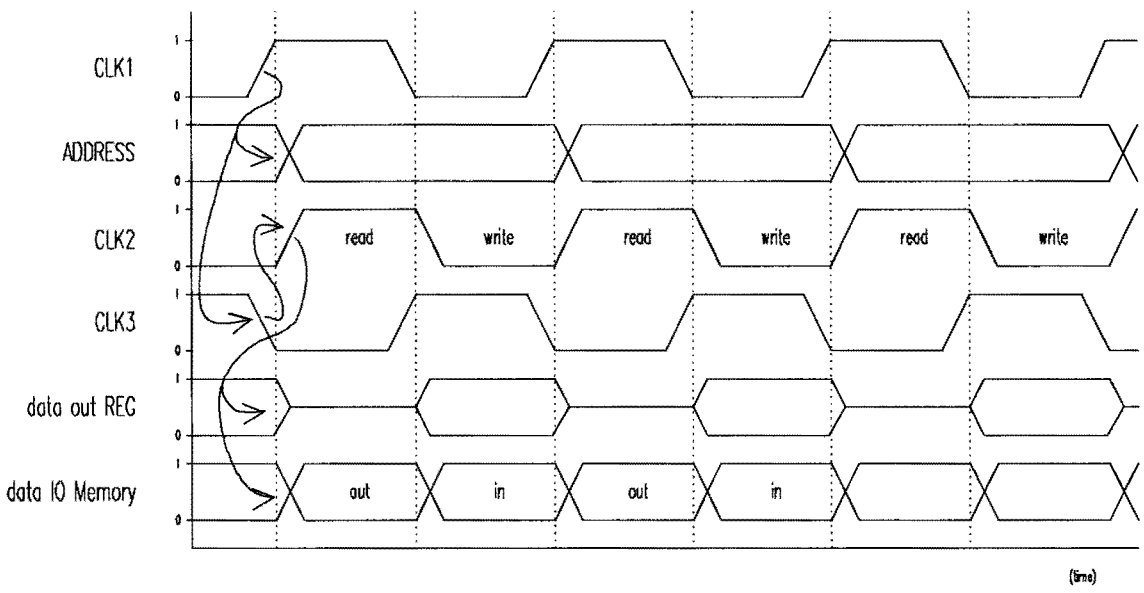



**Figure 18**: line buffer timing diagram to enable *READ MODIFY WRITE*

generator which will be active at the beginning of each video line. DATA[0..7] represents the 8 bit image input. The timing required for a READ MODIFY WRITE cycle is shown in Figure 18. The actual difference between the 3 clocks is a fixed phase shift. This shift is determined by the setup time of the address generator (stable address) and the propagation delay of the used registers. The circuitry providing the timing signals is included in the appendix.

READ MODIFY WRITE cycle:

Both memory cycles (read and write cycle) are WE (Write Enable) controlled. This means, in this case, when WE is low the memories can be written an when WE is high the memories can be read. CLK1 is used to generate the address, CLK2 is used as read and write signal on the memories and to enable the outputs of the registers (OE). CLK3 is used to latch new data in the registers. Since WE and OE are level signals and CLK of the registers is edge triggered, the following will happen:

* On the rising edge of CLK1 the new address will be generated. The outputs of the registers are disabled.
* Before CLK2 goes low, CLK3 goes high and the data that is on the data bus and on the outputs 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.

**Notice** that the reading and writing takes place in only one clock cycle, so the address does not chance. The result of this is that when data is read from the memory it will be overwritten in the second part of the clock cycle.

The main result with this approach will be that: At the outputs of the registers will appear new data every clock cycle.

Because the OPTIC filters need 8 bytes input data we will use a cascade of 8 memories in the configuration of Figure 17. The actual circuitry is contained in the appendix.

**Template Ram:**

The template ram is used for storage of feature templates as well as shape templates. The feature templates are used in the OPTIC filters and the shape templates in the IRIS filters. The organisation of the memory is quite open, both template types can be stored arbitrarily at any place in the 8k memory, as long as the templates start at an address that is a multiple of 16, so the 4 LSB's: 0000. As a result of this the memory could be mapped as is shown in Figure 19. A detailed map of an OPTIC template is shown in Figure 20. For

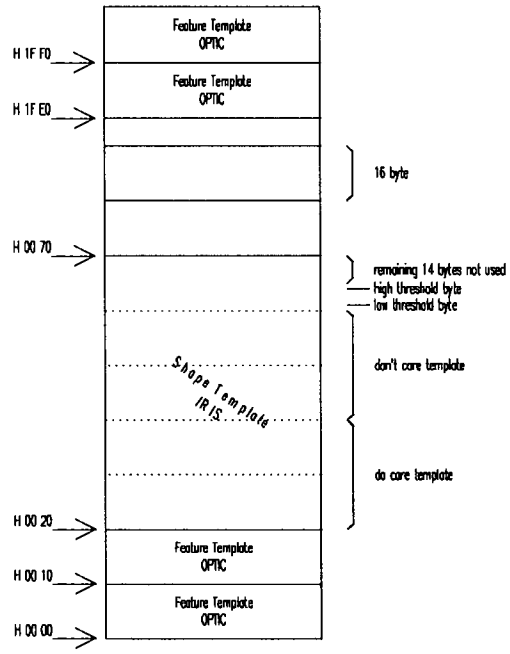



**Figure 19**: Template Ram memory map

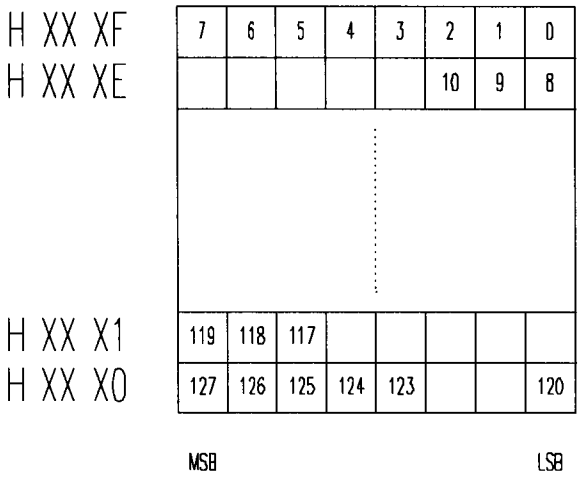



**Figure 20**: OPTIC template map

further details on the OPTIC see the chapter on the OPTIC. As can be seen in Figure 20 the range threshold for the OPTIC is not stored with the template in this memory. The purpose not to do so is because all four templates in an OPTIC filter use the same threshold (only 1 value can be programmed)

Reprogramming of the threshold value remains possible but this is only possible through the 22 bit shift register (Figure 21) which is also used to program the mode in which the OPTIC will work, the loading command for this data needs to come from the CAD reference image. The mapping of an IRIS template is shown in Figure 19 but for more details see the special chapter on the IRIS filter. Dependent on the application (kind of images to process) one can select his own set of templates and download them to this memory. The mechanism that selects the right template from this memory to be loaded in to one of the filters, will be discussed in the chapter concerning the CAD Control Logic. This memory will be mapped into the microvax memory for verification purposes. The loading of the templates onto the IRIS's and OPTIC's require a special control unit shown in Figure 21.

From Table II one can see that the Load Control (Figure 21) can take one of 3 main actions:

1. loading a template in one of the OPTIC's
2. loading a template in one of the IRIS's
3. load all the OPTIC's with the same configuration data.

Since an OPTIC needs to be loaded serially and the template is stored byte wise in the template ram, a parallel to serial conversion has to take place. For this reason the Load Control is equipped with a state machine that takes care of the addressing of the template ram, loading of the shift register and shifting the data out serially into an OPTIC. Figure 22 shows the timing diagram belonging to one shift cycle. After a START signal, the state machine will start generating the first address of the template indicated by the destination code and load the first byte into the shift register and starts shifting it out. Depending on the destination code a START signal will initiate a load cycle for an OPTIC template (16*8 + 16 = 144 clock

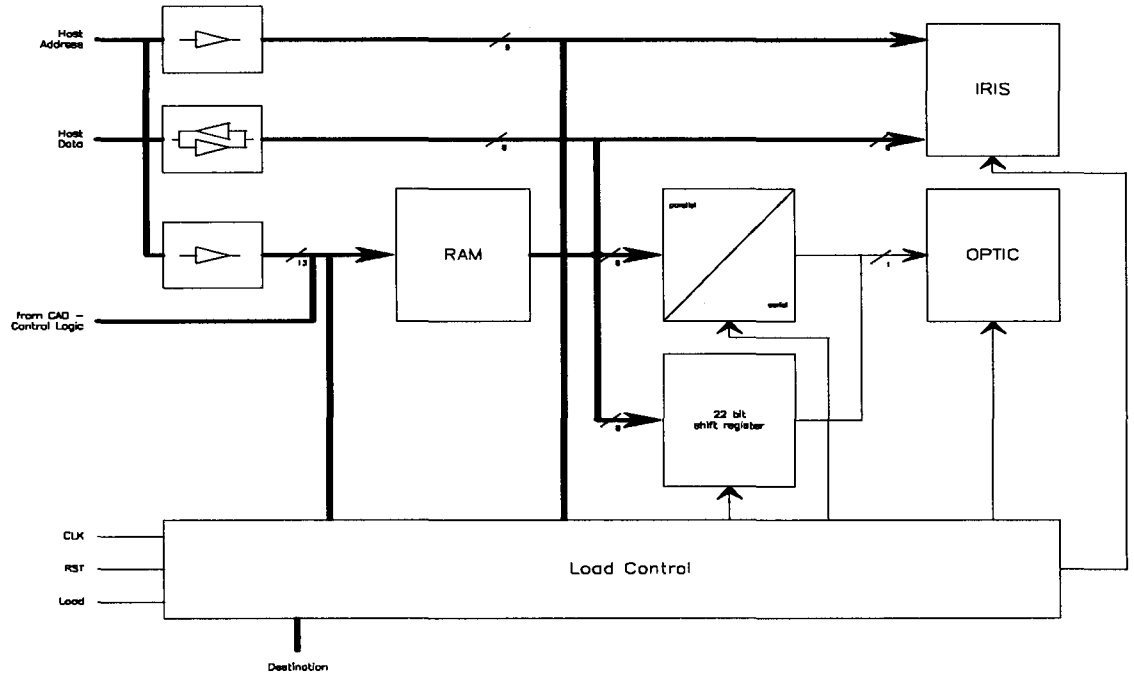



**Figure 21**: block diagram for load mechanism

cycles) or loading configuration data (22 clock cycles) into the OPTIC's. If the destination code indicates an IRIS template, the Load Control needs only to generate the appropriate addresses since an IRIS is loaded parallel. For more specific data on the loading of an OPTIC or an IRIS see the corresponding chapters.

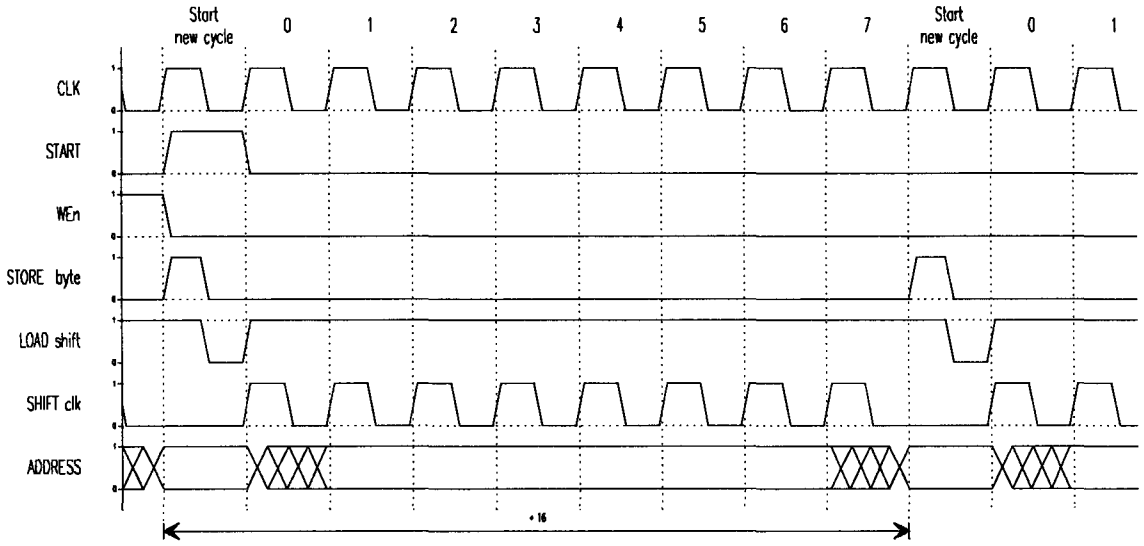



**Figure 22**: timing of 1 shift cycle

For detailed circuitry description see the appendix.

**Table II**: Destination codes for loading templates and configuration data of OPTIC's

| Destination | | Code | Destination | | Code |
|---|---|---|---|---|---|
| OPTIC: | section: | | OPTIC: | section: | |
| 0.0 | 0 | 0 00000 | 2.0 | 0 | 0 10000 |
| | 1 | 0 00001 | | 1 | 0 10001 |
| | 2 | 0 00010 | | 2 | 0 10010 |
| | 3 | 0 00011 | | 3 | 0 10011 |
| 0.1 | 0 | 0 00100 | 2.1 | 0 | 0 10100 |
| | 1 | 0 00101 | | 1 | 0 10101 |
| | 2 | 0 00110 | | 2 | 0 10110 |
| | 3 | 0 00111 | | 3 | 0 10111 |
| 1.0 | 0 | 0 01000 | 3.0 | 0 | 0 11000 |
| | 1 | 0 01001 | | 1 | 0 11001 |
| | 2 | 0 01010 | | 2 | 0 11010 |
| | 3 | 0 01011 | | 3 | 0 11011 |
| 1.1 | 0 | 0 01100 | 3.1 | 0 | 0 11100 |
| | 1 | 0 01101 | | 1 | 0 11101 |
| | 2 | 0 01110 | | 2 | 0 11110 |
| | 3 | 0 01111 | | 3 | 0 11111 |
| IRIS: | section: | | IRIS: | section: | |
| 0 | 0 | 1 00000 | 1 | 0 | 1 00100 |
| | 1 | 1 00001 | | 1 | 1 00101 |
| | 2 | 1 00010 | | 2 | 1 00110 |
| | 3 | 1 00011 | | 3 | 1 00111 |
| Loading OPTIC's with configuration data: | | | | | |
| OPTIC: | section: | | | | |
| ALL | ALL | 1 10000 | | | |

### Override register

A override register is provided. This register can be used to store override data that is generated by the host computer for testing purposes.

At the time of this report the CDCX2 (Cad Data Correction and eXpansion unit 2) which has to generate the CAD input image for the SFMU (Shape Feature Measurement Unit) is not build yet. To be able to test the SFMU we need to provide an input image. Since the picture busses of the PAPS system are only 8 bits wide, we can use several of these busses or let the host computer generate a reference image.

Using one or more picture busses will be used to do more or less realistic tests with the board for as the host computer will be used to generate the more elementary instructions like loading thresholds, modes and this sort of thinks.

The host computer will always be master, this means the host computer is the only one that can determine if the host computer itself or the CDCX2 has control over the SFMU. This is done by programming a single bit register to 1 (host control) or to 0 (CDCX2 control). This bit will be used to switch a multiplexer that chooses between the override register or the CDCX2 input port. A block diagram of this is given in Figure 23.

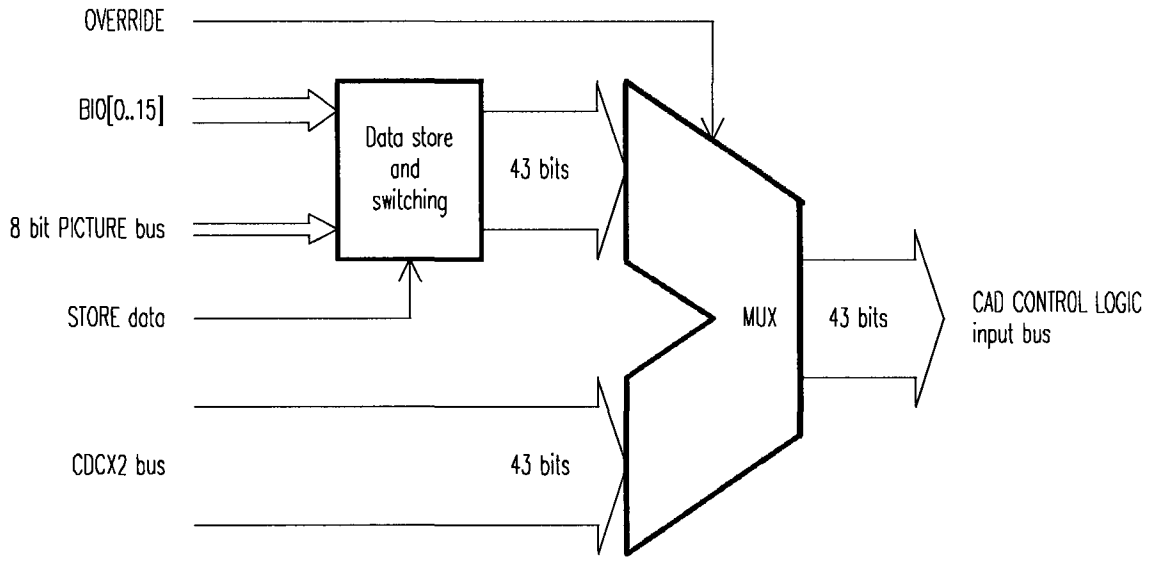



**Figure 23**: block diagram of override mechanism

### CAD Control Logic

The CAD Control Logic represents the main circuit controler. It controles the template ram, the offset between the grids, the OPTIC filters, stored template ram for the OPTIC's, the IRIS filters and the stored template ram for the IRIS's.

For the template ram, it has to provide the base address pointer for the templates to be loaded when this is requisted by the CAD input data.

It has to enable the CAD input data specified templates in the different filters or load one of them with a new template or load all OPTIC's with a new threshold. While loading a new template in an OPTIC or an IRIS, it has to update the coresponding stored template ram for output purposes.
An other important feature is the offset between the grids. All four grids are supplied with the same input image but the enable windows of the templates are shifted with respect to each other (8 pixel in both horizontal and vertical direction). This means providing the right timing signals for the different grids.

From these conciderations we come to the following input format for the controler (output of the CDCX2).

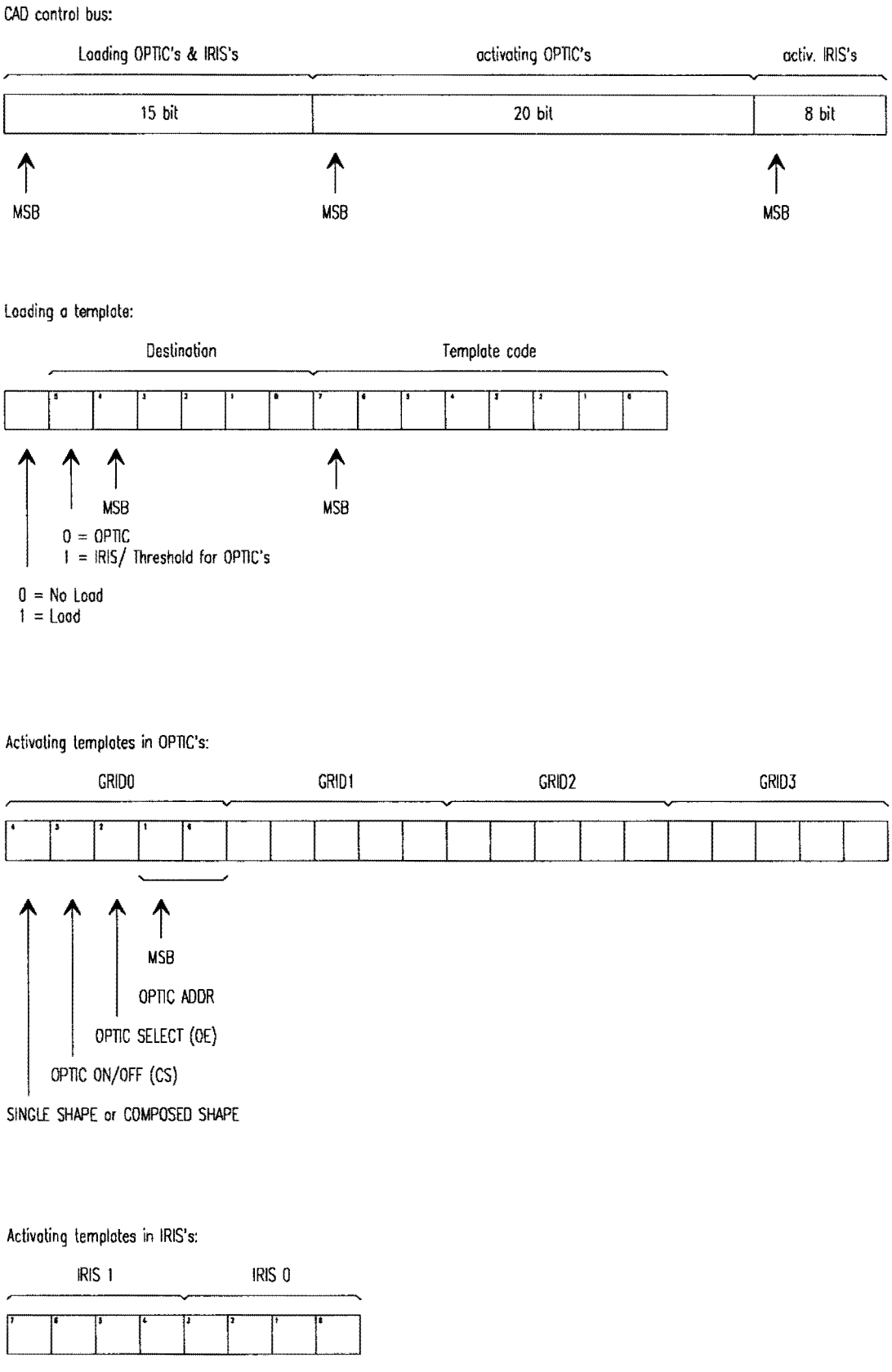



**Figure 24**: Input format description of Shape Feature Measurement Unit

From Figure 24 we see that the CAD control bus needs to be 43 bits wide and is composed of 3 main busses:

1. Loading OPTIC's and IRIS's
2. Activating OPTIC's
3. Activating IRIS's

Table II shows the destination codes as they are being used by the CAD Control Logic

**Loading the templates:**

Loading a template to the appropriate position in the right filter, is done by specifying the destination and the template code and setting a "Load"-bit (MSB of "Loading a template"-part of Figure 24). The destination codes are given in Table II. This procedure is valid for both OPTIC and IRIS filters and more important the loading can be done off grid.

The template code is an unique user defined 8 bit code, labelling the used templates. OPTIC-templates and IRIS-templates must have different codes. So 256 template codes are available.

**Activating the templates:**

OPTIC's:

Every grid uses 2 OPTIC filters, capable of containing 4 templates each, so every grid can switch between 8 templates. To address one of them a 3 bit address has to be provided, this is indicated by bit 2,1 and 0 in Figure 24, the "Activating templates in OPTIC's:"-part. Bit 3 is used to turn both OPTIC's on (bit 3 = 1) or off (bit 3 = 0). Finally bit 4 indicates if a single feature is regarded as a single shape (bit 4 = 0) or as part of a composed shape (bit 4 = 1). Incase of single shape the second layer will be bypassed and the result of the first layer will go directly to the output logic.
This construction is valid for all four grids.

IRIS's:

In the second layer, 2 IRIS chips will be used, each of them partitioned in 2 groups of 2 16*16 templates. So each IRIS contains 4 templates referenced to in Table II as sections 0 through 3. For grid 0, section 0 and section 1 of IRIS 0 will be used. For grid 1 the sections 2 and 3 of IRIS 0 will be used. Likewise IRIS 1 serves grid 2 and 3 respectively. An 1 indicates an active template and a 0 an inactive template.

CAD Control Input LUT

To be able to provide the appropriate base pointer to the template ram, a translation needs to be made from the template code to the physical address in the template ram. For this purpose a LUT (Look Up Table) is provided. So the template code comming from the CDCX2 will be used as an addres for this memory the output of this memory (the data bus) will be connected to the template ram's address bus. This LUT will also be mapped in the microvax memory for testing and programming reasons.
The user has to take care that de user defined template codes refer to the correct base pointer address on the template ram, so the contents of the LUT.

### Grid0..3

Figure 16. shows the construction of the first layer being composed of 4 grids and a Stored Template Ram. This chapter will discuss the Grid Offset between the grids, give an extensive overview of how to and what the OPTIC's can do and will be doing and last the 2k Line Buffer as output stage of a grid.

### Grid Offset

To be able to use the proposed grids, it is necessary to provide each grid with its own offset. This can simply be done by suppressing the clock signals to the grids for an appropriate period of time. Figure 25 shows a total image frame with the starting points of the grids (0, 1, 2 and 3 corresponding to Grid0, Grid1, Grid2 and Grid3) and along side the frame the timing signals for line- (LIE0 and LIE1) and picture- (PIE0 and PIE1) enable. PIE0 equals PIE and LIE0 equals LIE. PIE1 is a Grid Offset delayed version of PIE, likewise LIE1 is a Grid Offset delayed version of LIE.

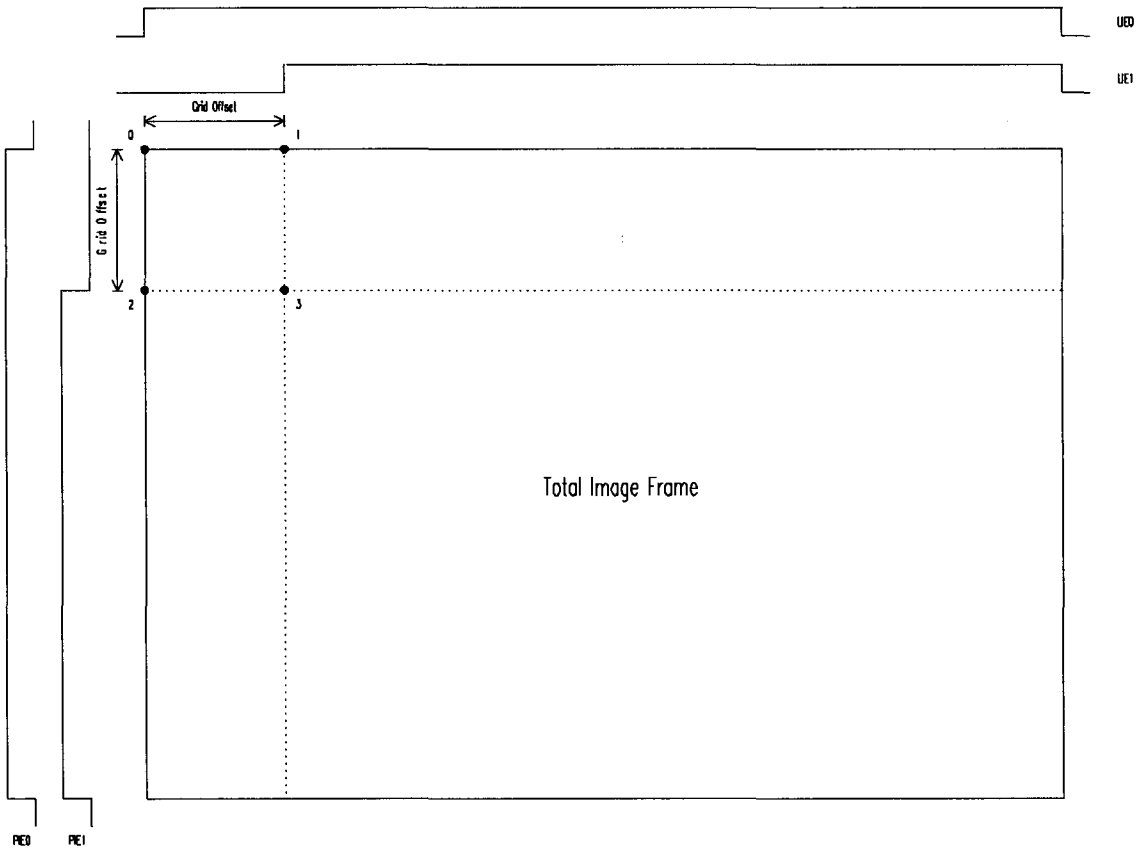



**Figure 25**: Grid enable signals

From these signals one can easily extract the logic functions that can be used as enable signals for the different grids. Equation (16) lists these functions.

$$\begin{aligned} Grid0 &= LIE0 \cdot PIE0 \\ Grid1 &= LIE1 \cdot PIE0 \\ Grid2 &= LIE0 \cdot PIE1 \\ Grid3 &= LIE1 \cdot PIE1 \end{aligned} \tag{16}$$

By using these enable signals to gate the clock signals for each grid the grids will in time and therefor in place be shifted with respect to each other. The offset will be fixed to 8 pixels horizontal and 8 lines vertical these values are chosen to be able to detect features that are less then 16 pixels (grid size) apart.

**OPTIC** Optimised Pixel Template Image Correlator

The OPTIC is a VLSI circuit by S3 that computes the maximum and minimum of a set of 64 8 bit unsigned pixels at a pixel sampling rate up to 20MHz. The architecture of this IC is shown in Figure 26.

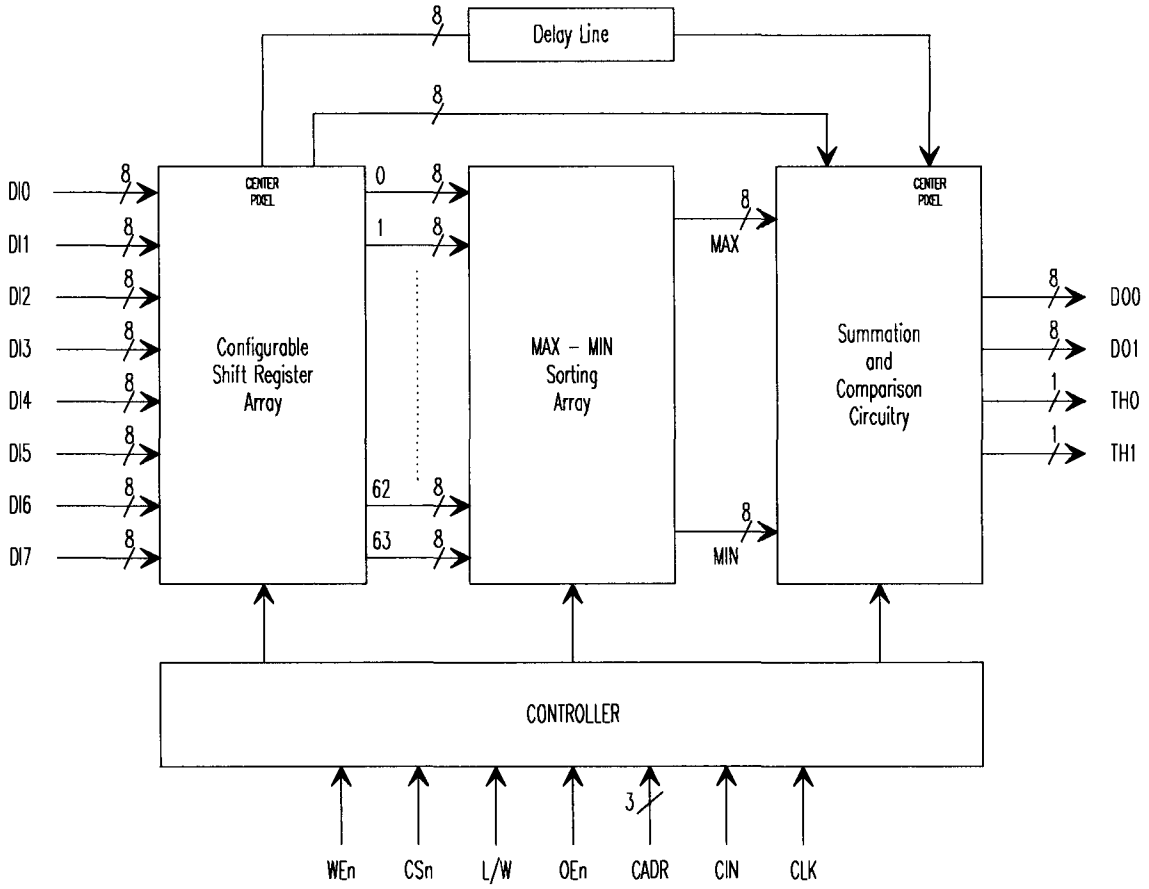



**Figure 26**: block diagram of OPTIC architecture

A configurable shift register array allows the 64 pixels to be arranged in on of the following window configurations, Table III.

**Table III**: Window shapes

| Window shape | C[21..18] | C[3..0] | Window inputs |
|---|---|---|---|
| 8 * 8 | 1111 | 1111 | DI7 → DI0 |
| 4 * 16 | 0101 | 0101 | DI6, DI4, DI2, DI0 |
| 2 * 32 | 0001 | 0001 | DI4, DI0 |
| 1 * 64 | 0000 | 0001 | DI0 |

The ranking array sorts the pixels to find the maximum and minimum values. A key feature of this device is that the pixels may be arbitrarily divided into two groups, one for maximum and minimum calculations only. This feature is necessary to allow the implementation of Multi-level Input Binary Template Matching (Dynamic Range Correlation). Alternatively all pixels may be considered for maximum and minimum calculation as in a normal image filter. In both modes pixels may be arbitrarily masked from the ranking process. This part is referenced by MAX-MIN Sorting Array in Figure 26. The possible attributes for each pixel in the window is listed in Table IV.

**Table IV**: Possible pixel attributes

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

The actual sorting is done by an pipelined (6 deep) tree of comparators configurated as is shown in Figure 27

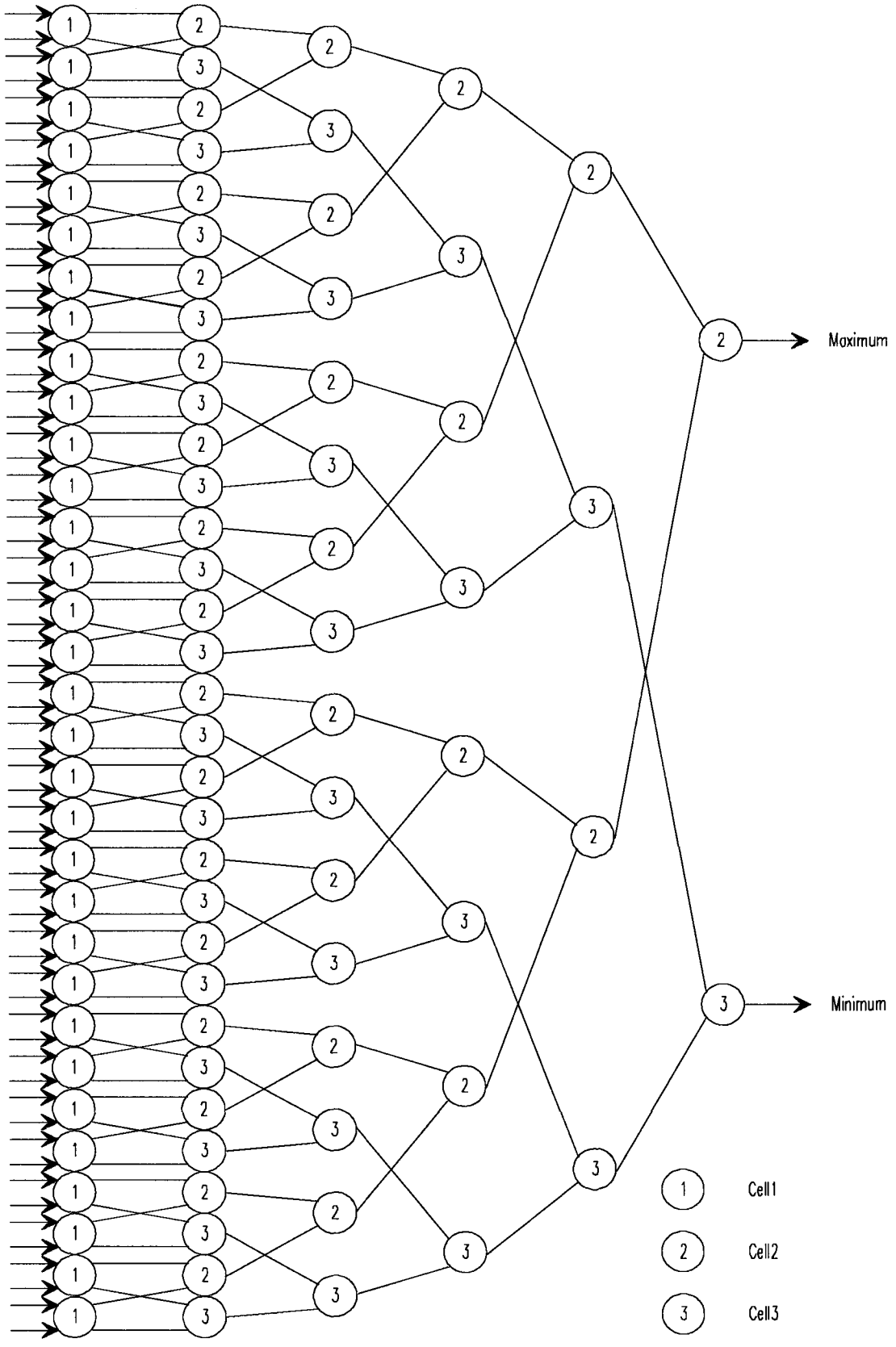



**Figure 27**: comparitor tree of sorting array

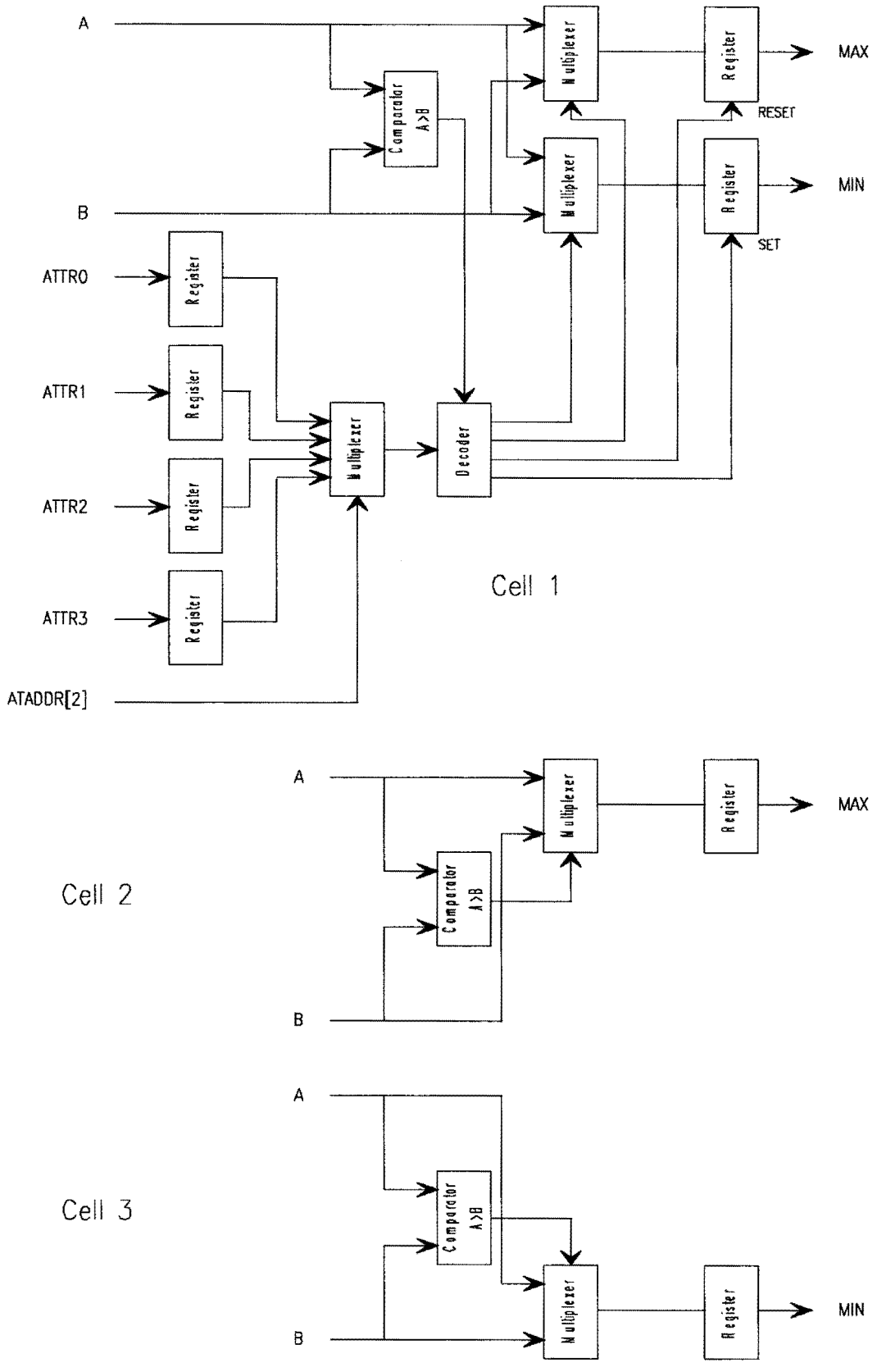



**Figure 28**: sorting cell architecture

The cells in Figure 27 represent the more detailed architecture of the corresponding cells in Figure 28. Depending on the programmed pixel attribute cell takes one of the actions shown in Figure 29.

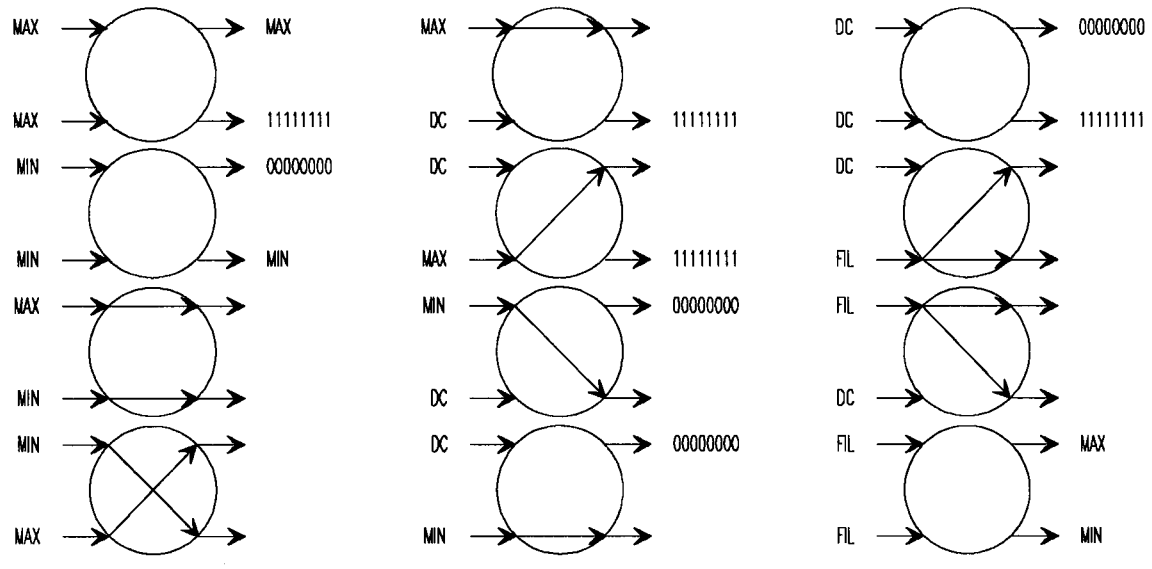



**Figure 29**: cell 1 decision rules

As can be seen from Figure 28 cell 1 will use, depended on the ATADDR[2] (address), one of four programmed pixel attributes. By choosing one of them, one of the four programmed templates is chosen.

The summation and comparison circuitry indicated in Figure 26 combines the maximum and minimum values calculated in the sorting array and routes them to the filter outputs according to the MODE chosen, Table V lists the supported modes. The center pixel of CP used is the pixel loacated at register 4,4 in the Shift Register Array delayed the same number of clock cycles as the sort array i.e. 6 cycles.

**Table V**: supported operation modes

| C[15..13] | DO0 | DO1 |
|---|---|---|
| 000 | MIN | CPNF |
| 001 | (MAX + MIN)/2 | CPNF |
| 010 | MAX | CPNF |
| 011 | DIFF | CPNF |
| 100 | MIN | MIN |
| 101 | (MAX + MIN)/2 | (MAX + MIN)/2 |
| 110 | MAX | MIN |
| 111 | DIFF | (MAX + MIN)/2 |

DIFF and CPNF are defined as follows:

DIFF = (MIN - MAX) if c[4]=0
DIFF = (MAX - MIN) if c[4]=1

CPNF = Center Pixel if c[16]=0
CPNF = Noise Filter if c[16]=1

The 2 outputs TH0 and TH1 are determined as follows:

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

The Control logic in Figure 26, controls the operating modes, LOAD mode or WORK mode, controllable by the L/W input. In the load mode all control data is downloaded into the filter, i.e. all window shapes, mode, pixel attributes and threshold data. This data is loaded serially through the CIN input under control of the WEn signal. This transfer is synchonous to CLK. The control bus, CADR, determines which data chain is loaded. The filter outputs are in tri-state in LOAD mode and OEn is disabled. Changing to WORK mode fixes all control data and WEn is disabled. The outputs are controlled by OEn. In WORK mode the 2 LSB bits of CADR determine which set of pixel attribute data is used. A ship select (CSn) is also available.

Loading of the templates and configuration data:

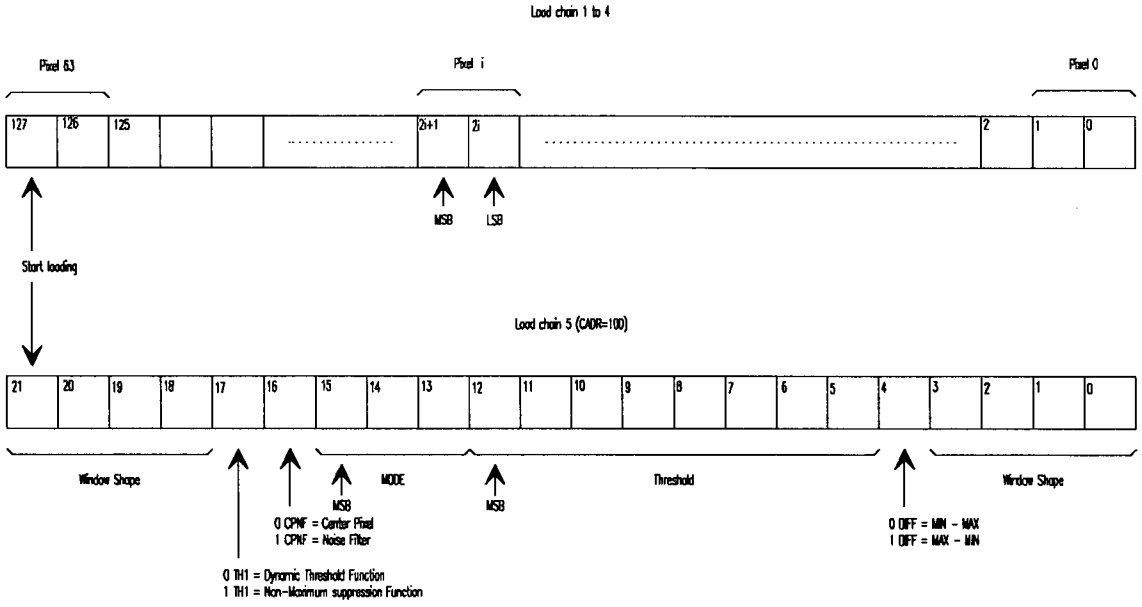



**Figure 30**: Construction of load chains for OPTIC processor

So there are 5 chains to be loaded, 4 template chains and 1 configuration chain.
The construction of the load chains is depicted in Figure 30, the different chains can be addressed using CADR as listed in Table VI.

**Table VI**: Template load chain addressing

| CADR | Control Chain Selected |
|---|---|
| 000 | template 0 |
| 001 | template 1 |
| 010 | template 2 |
| 011 | template 3 |
| 100 | window shape, etc. |

The template chain as indicated in Figure 30 can be mapped on a 8*8 template as is indicated in Figure 31.

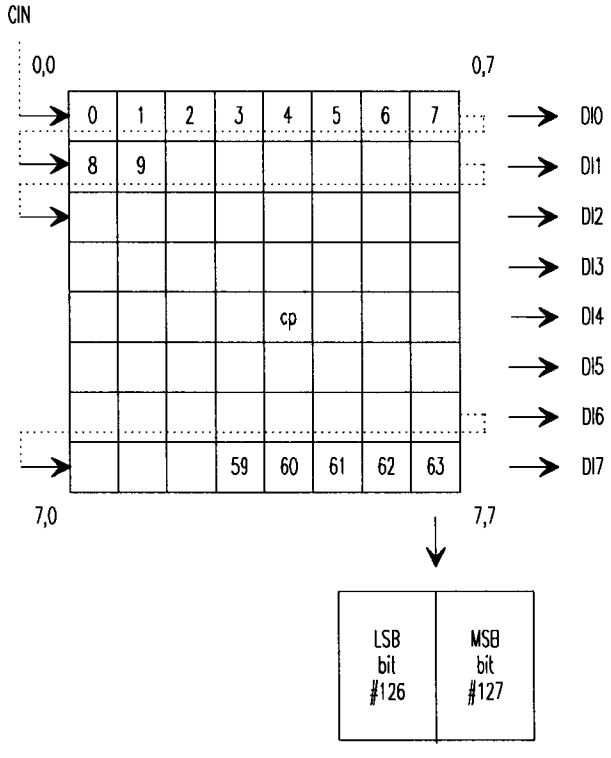



**Figure 31**: template chain mapped on 8*8 window

To be able to use the filter as a Multi-level
Input Binary Template Matcher for an 8*8 window, the configuration chain would be constructed like Figure 32.

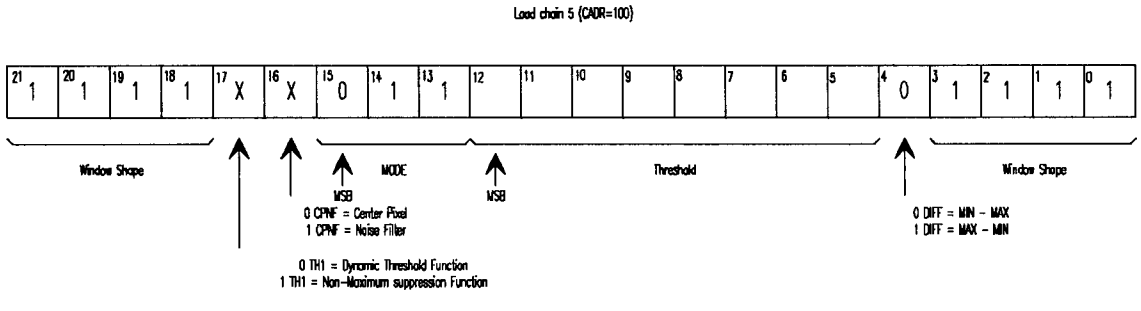



**Figure 32**: configuration chain in case of Multi-level Input Binary Template Matching

Where the threshold can have an user defined value.

## 2k Line Buffer

The 2k (1/16 of normal line length) line buffer consists of 2 parts, the first one a data reduction unit and seconde a conventional line buffer for 16 lines (binary). Figure 33 shows this construction.

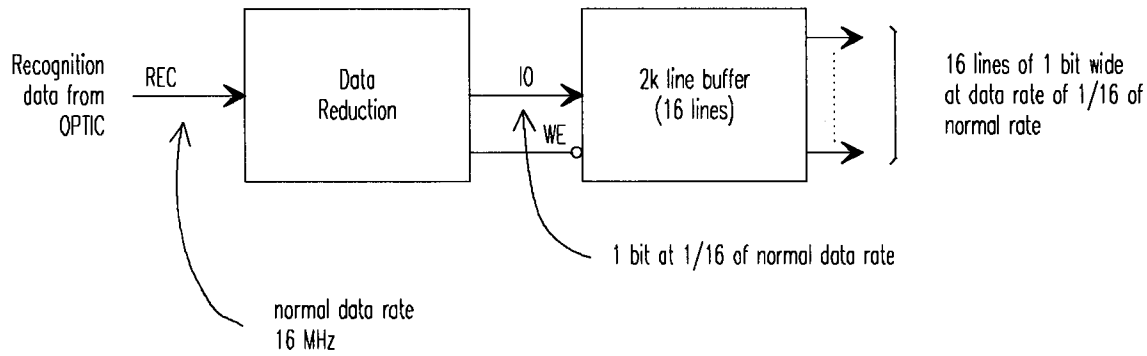



**Figure 33**: block diagram of 2k Line Buffer

## Data reduction

General propagation of recognition point:

Figure 34 shows a magnified part of one grid placed in an image. Here the grid size is taken to be 4 this is just for clarity. When a recognition takes place in a grid cell (First Recognition) this data (1 bit) is written in a memory, since the address on this memory will be the same throughout the cell, later recognitions will also be stored at the same address (overwritten). It will be assumed that the recognition data will be synchronous with CLK2.

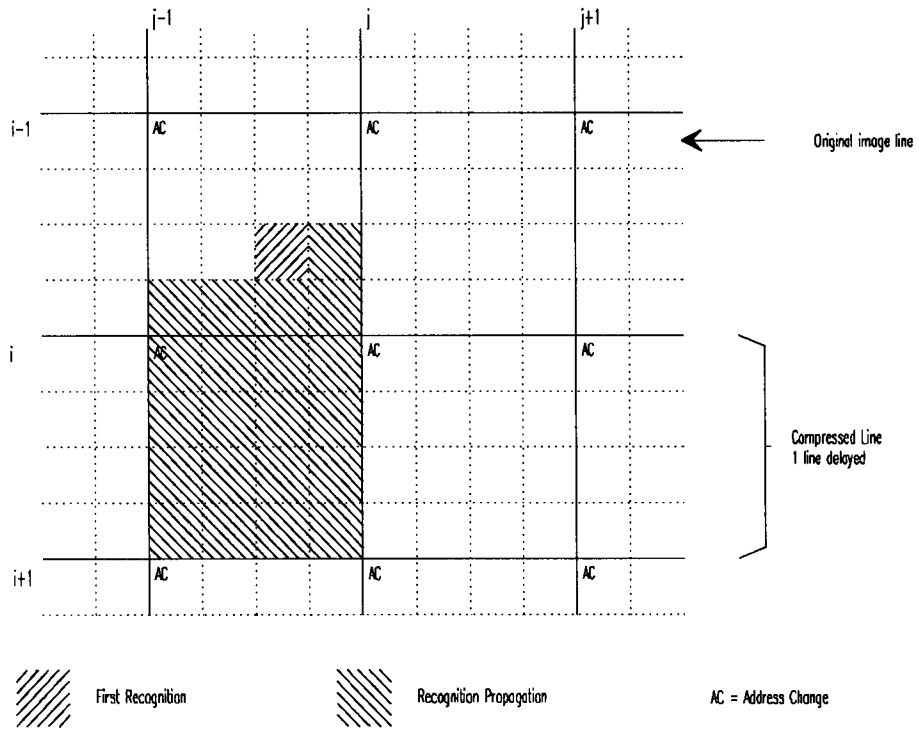



**Figure 34**: recognition propagation

During normal operation (GRID=0 this means you are in a cell and not on de boundaries) the following timing diagram is valid:

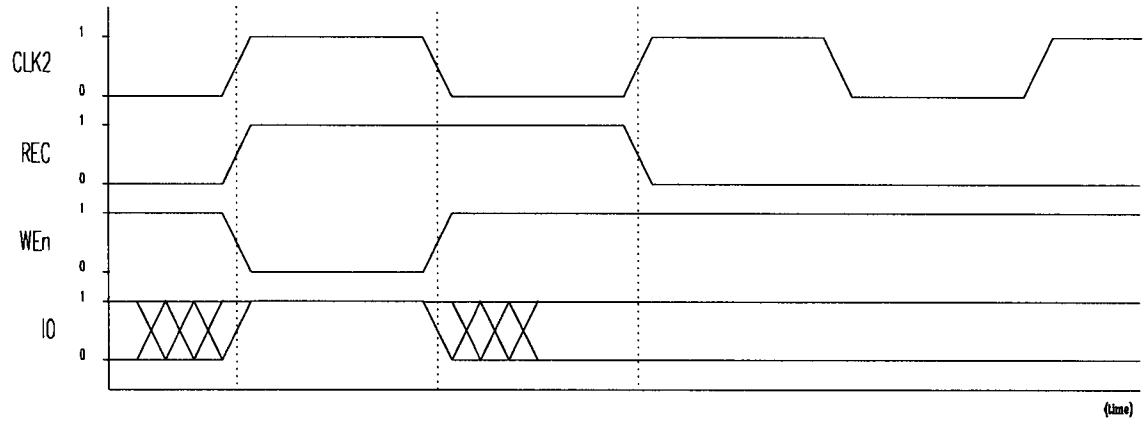



**Figure 35**: normal operation timing of recognition capture

From this timing diagram one can extract to following logic equation:

$$\begin{aligned} GRID &= 0 \\ I0 &= REC \\ \overline{WE} &= \overline{CLK2} + \overline{REC} \end{aligned} \tag{17}$$

Leading to the following equations:

$$\begin{aligned} I0 &= REC \cdot \overline{GRID} \\ \overline{WE} &= (\overline{CLK2} + \overline{REC}) \cdot \overline{GRID} \end{aligned} \tag{18}$$

During operation on grid (GRID=1 AC in Figure 34) the following timing diagrams are valid:

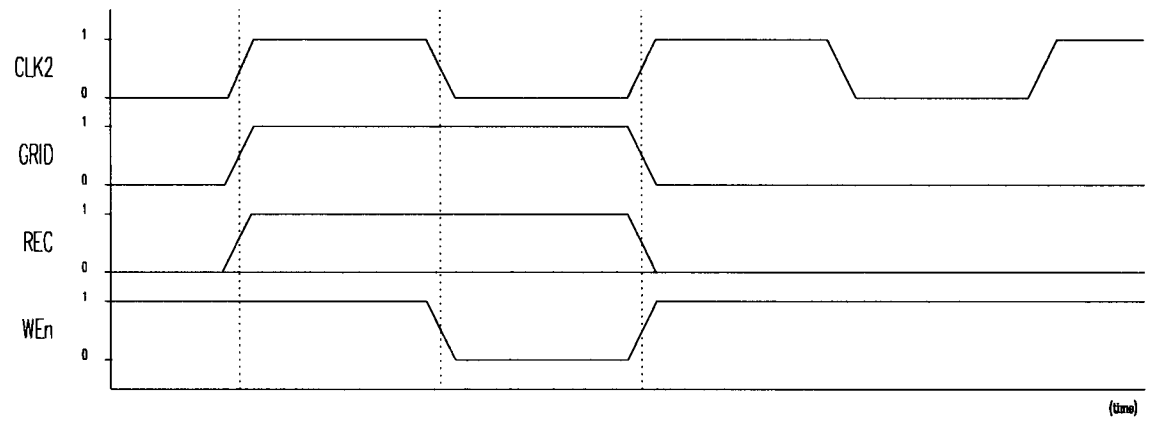



**Figure 36**: recognition capture on grid (Address Change)

From this diagram one can extract the following logic equations:

$$\begin{aligned} GRID &= 1 \\ I0 &= REC \\ \overline{WE} &= CLK2 + \overline{GRID} \end{aligned} \tag{19}$$

Leading to the following equations:

$$\begin{aligned} I0 &= REC \cdot GRID \\ \overline{WE} &= (CLK2 + \overline{GRID}) \cdot GRID \\ &= CLK2 \cdot GRID \end{aligned} \tag{20}$$

Combining these sets of equations leads to the following set:

$$\begin{aligned} I0 &= REC \\ \overline{WE} &= \overline{GRID} \cdot (\overline{CLK2} + \overline{REC}) + GRID \cdot CLK2 \end{aligned} \tag{21}$$

Leading to the hardware interpretation of Figure 37.

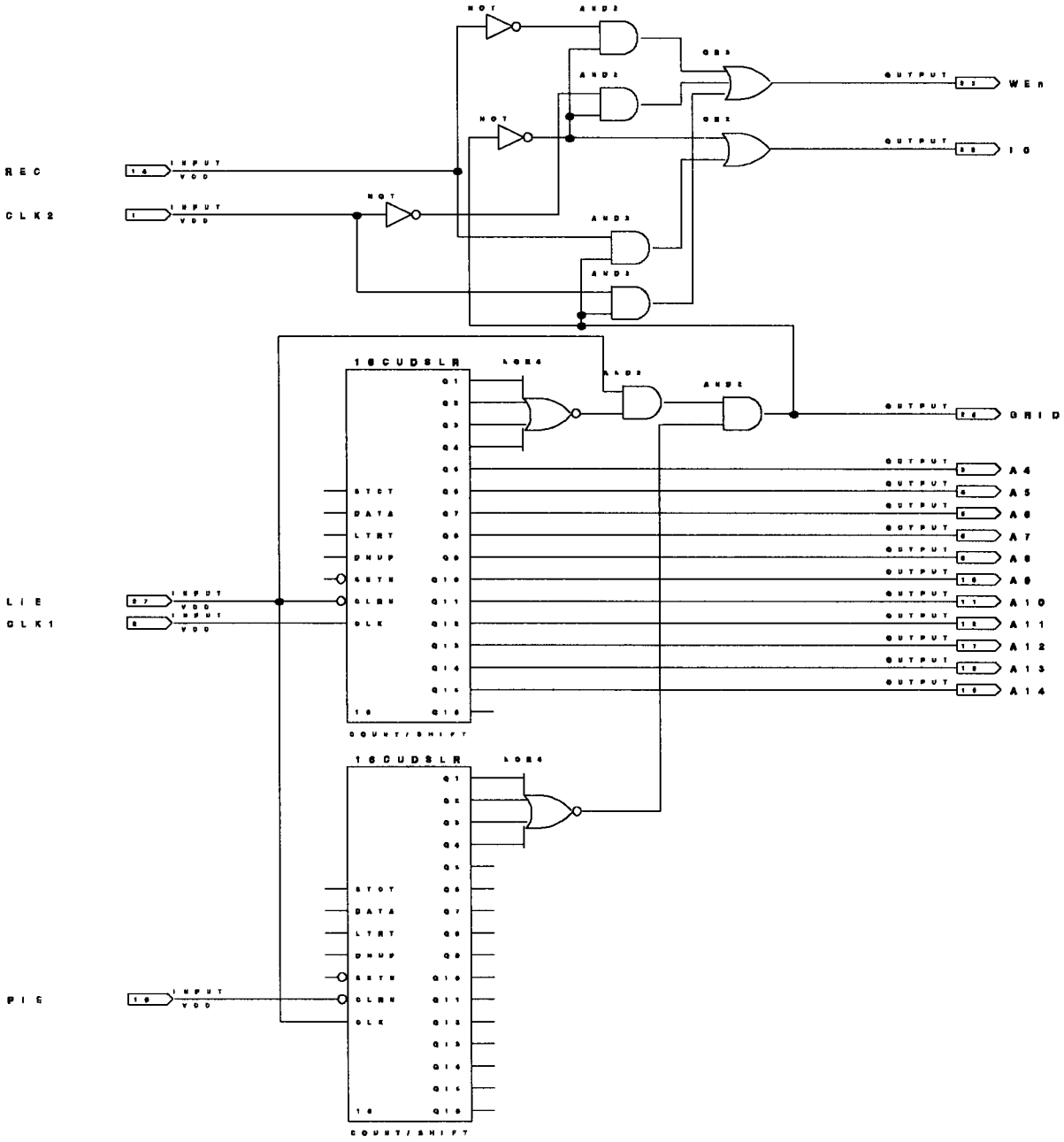



**Figure 37**: hardware implementation of recognition capture

**16 lines binary line buffer**

The actual data reduction is preformed by a 16 lines, each 2k byte long, line buffer with the following timing diagram:

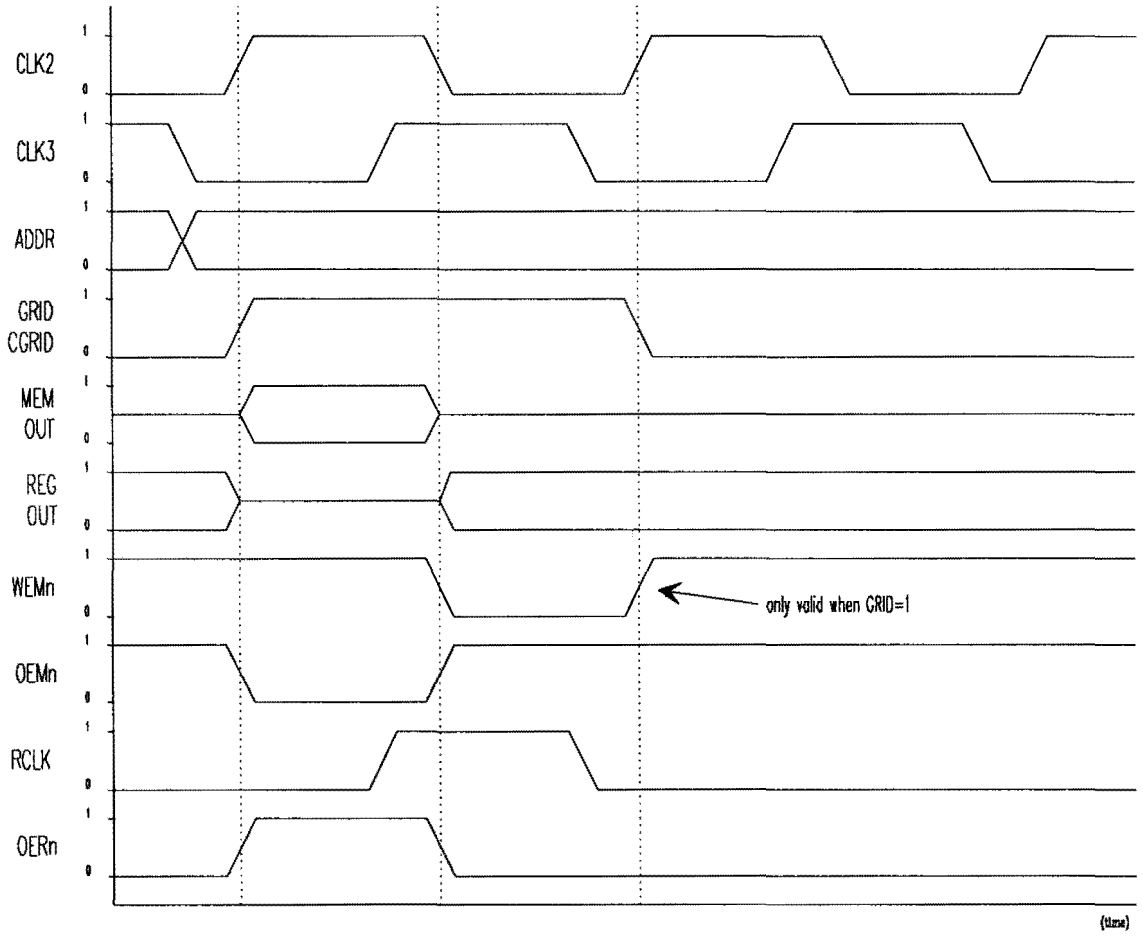



**Figure 38**: timing of actual line buffer control

One has to take into account that the data rate is now 1/16 of the normal in coming data rate which is normally 15 MHz.

From this timing diagram the following logic equations can be extracted:
WEMn : Write Enable Memory (active low)
OEMn : Output Enable Memory (active low)
RCLK : Register CLocK
OERn : Output Enable Register (active low)

$$\begin{aligned} \overline{WEM} &= \overline{GRID \cdot \overline{CLK2}} = \overline{GRID} + CLK2 \\ \overline{OEM} &= \overline{\overline{CGRID} \cdot CLK2} = \overline{CLK2} + \overline{CGRID} \\ RCLK &= CGRID \cdot CLK3 \\ \overline{OER} &= CGRID \cdot CLK2 \end{aligned} \qquad (22)$$

The accompanying circuitry can be found in the appendix.

**Stored Template Ram**

This small memory is used to keep track of which templates are stored in which OPTIC. It is necessary to store the template codes as they are input to the CAD Control Logic for loading, to be able to let the Output Logic know which output code needs to be generated in case of missing features of the first layer recognition. So with 32 feature templates stored the memory size is 32 bytes. This memory will also be mapped on the host computer for verification purposes. The CAD Control Logic takes care of the storing of template codes.

**IRIS** Image Recognition Integrated System

The IRIS is a single chip system of video line delays and high speed binary template matchers. The chip is switchable into 46 different configurations varying from a conventional 32*32 bit binary template matcher driven by 512 bit long line delays to 4 completely separate systems each containing two 8*16 binary template matchers in parallel, driven by 512 bit long line delays. In addition it can be configured into a 1-dimensional correlator. The line delay length is variable and the coefficients (template) can be changed in time to perform adaptive correlation. The block diagram belonging to the IRIS is shown in Figure 39.

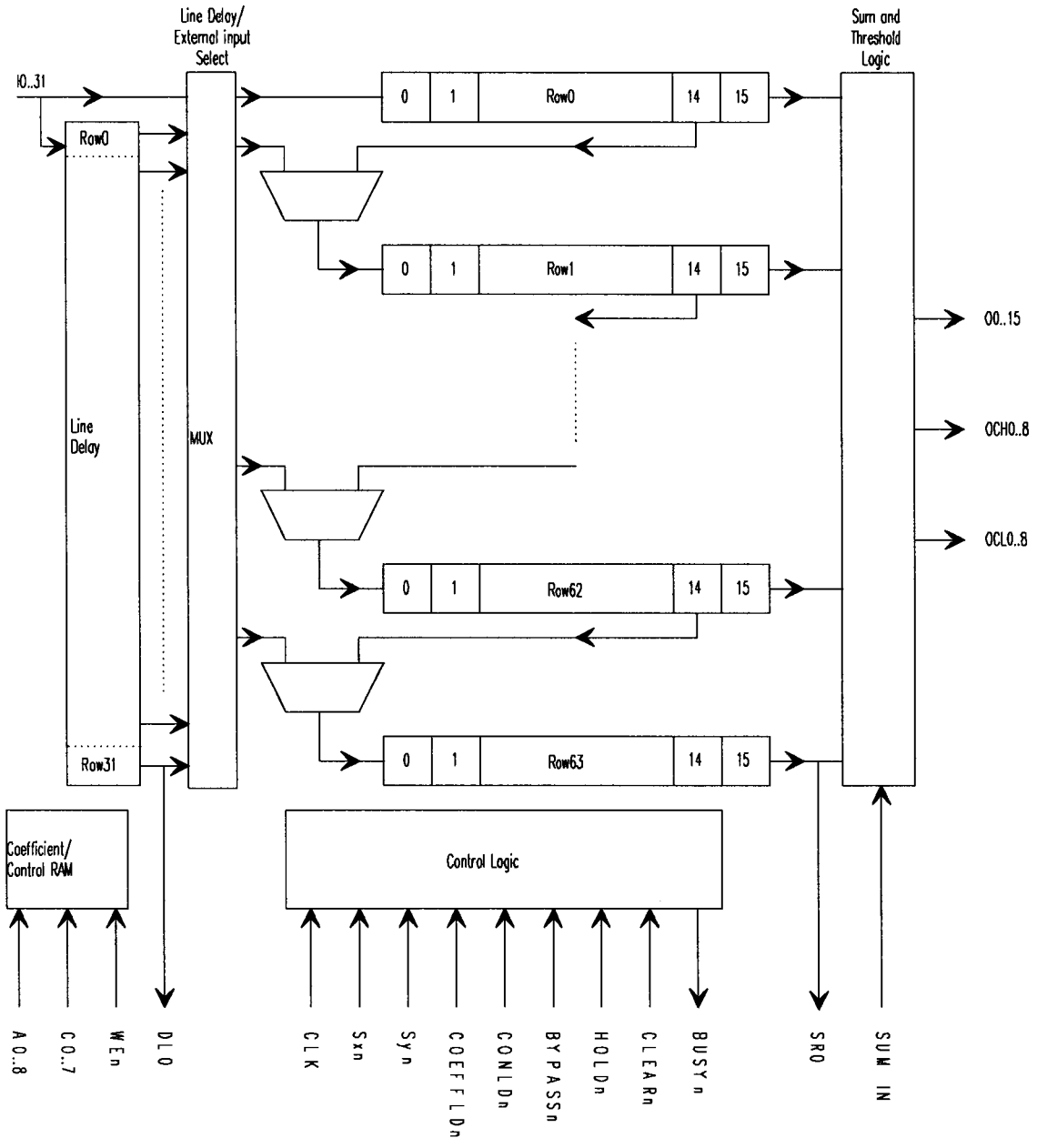



**Figure 39**: physical block diagram of the IRIS

Functional description

Figure 39 shows that the IRIS consists of 32 line delays driving 64 rows of 16 comparator cells. The result of the comparisons are summed and output (O0..O15). Output logic is added to clip (CLIP(A,B)=1 if A>B else 0) the summed results to single-bit outputs (OCH0..8 and OCL0..8). The multiplexer labelled MUX is used to bypass the line delays. The Coefficient/Control ram is used to store the templates and control data. This can be done asynchronously to CLK and in parallel with normal filter operation of the IRIS (IRIS can continue working). The templates can be down loaded into the comparator array using the COEFFLDn input, this takes 2600 clock cycles. Similarly the control data can be loaded by driving CONLDn low. This takes 200 clock cycles. DLO is a buffered output from the last delay line and SRO a buffered output from the last comparator cell row. These outputs may be used when expanding into a system using several chips.

IRIS modes

The IRIS can be switched to 46 different modes (46 different template shapes and partitioning, see preliminary data sheet of IRIS for all modes). In general each configuration can be thought of as having a

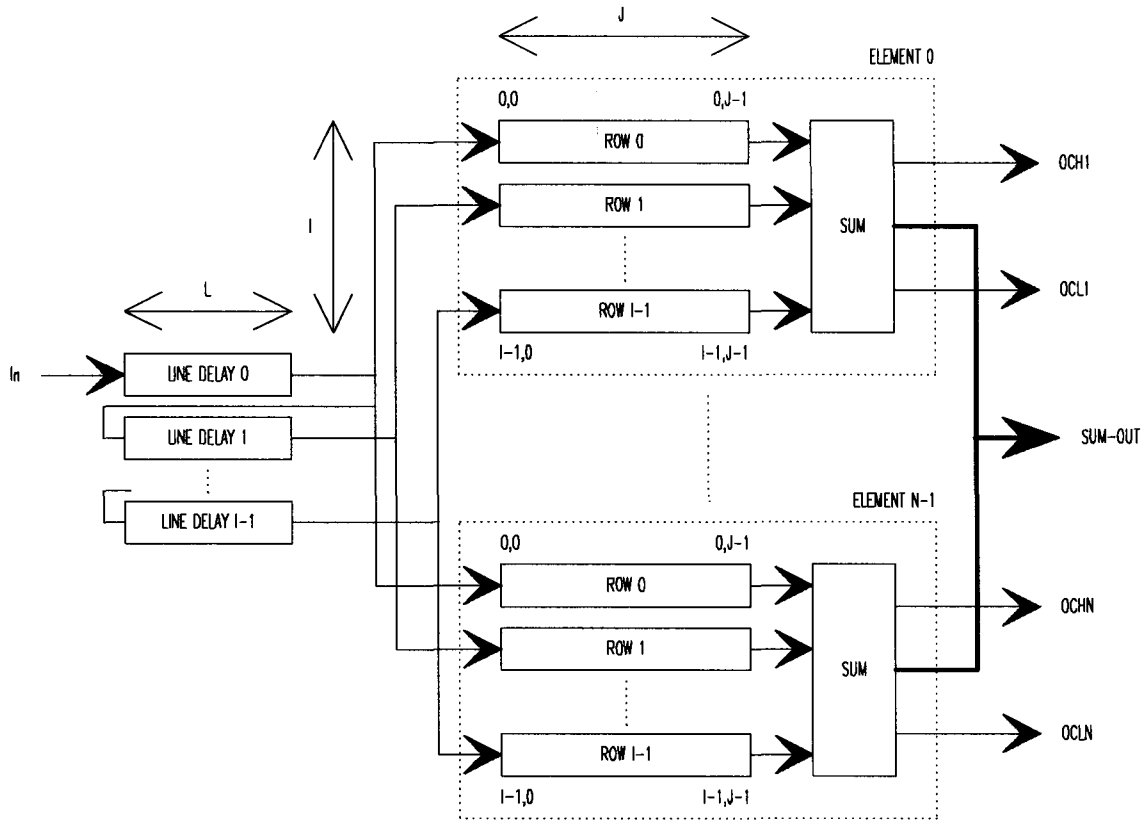



**Figure 40**: single system block diagram

form as is shown in Figure 40. As can be seen a system consists of N comparator (binary template matchers) arrays in parallel (N=1,2,4 of 8) driven by line delays with length L. The array consists of I rows of J comparator cells. The outputs from a system are the clipped values from each array as well as the total sum contribution from all the arrays (OCH0 and OCL0).

Figure 41 shows how an IRIS can be configured into M systems (M=1,2,4 or 8) . Independent on the number of systems used in one IRIS, one can use maximum 32 input bits, coming from the line delay or from an external source (as in our case). The sum outputs from each system are added together to form a final sum (O0..O15). This sum can be delayed by V (maximum 16) clock cycles to allow cascading of

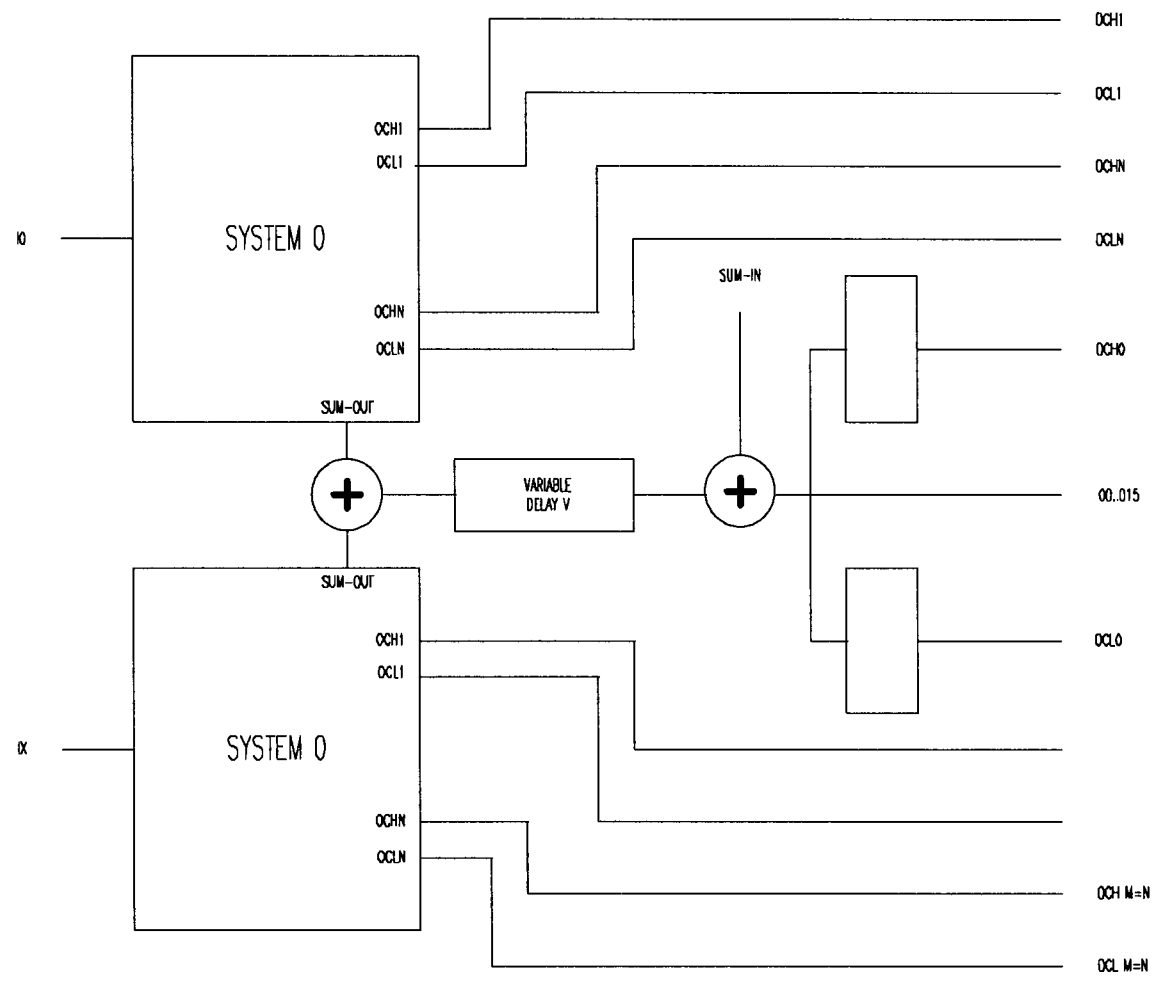



**Figure 41**: system configuration of IRIS

several chips. The delayed sum is added to a 16 bit sum input (SUM-IN) and output together with the clipped values of each system. The 16 bit sum output can also be clipped to 2 one bit values (OCH0 and OCL0). For timing and delay data see the preliminary data sheet of the IRIS.

**Operation**

Binary Template Matcher Array

As mentioned in the previous section IRIS can be configured in M systems of N Binary Template Matcher arrays. Each array consists of I rows of J comparator cells. The basic operation performed by each array is a windowed pixel by pixel comparison (binary convolution) between an user defined reference template and an input picture which is shifted into the array under control of the external clock. As well as a reference template there is also a "don't care" template which is used to mask off selected pixels from the comparison.

A block diagram is shown in Figure 42. Each of these comparator cells consist of a shift register element whose output connects to the next cell in a row. Each cell also contains 2 storage latches for storing the ref

$$cout = \overline{qn \oplus ref} \cdot \overline{dc} = qn \odot ref \cdot \overline{dc} \qquad (23)$$

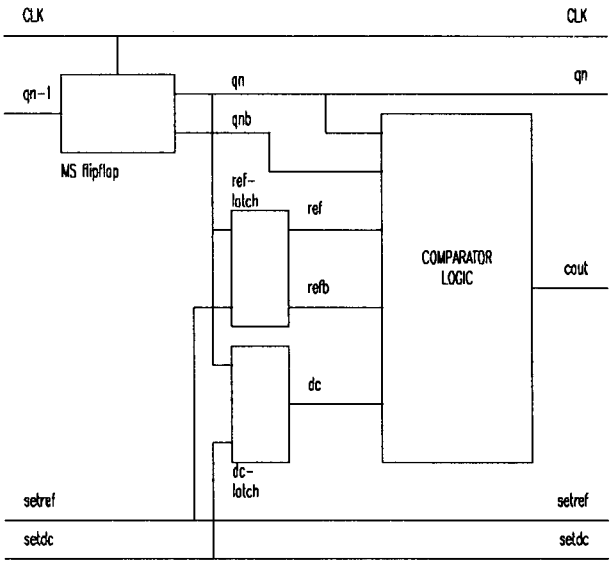



**Figure 42**: block diagram comparator cell

**Table VII**: Comparator cell logic output table

| dc | qn | ref | cout |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| | 0 | 1 | 0 |
| | 1 | 0 | 0 |
| | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| | 0 | 1 | 0 |
| | 1 | 0 | 0 |
| | 1 | 1 | 0 |

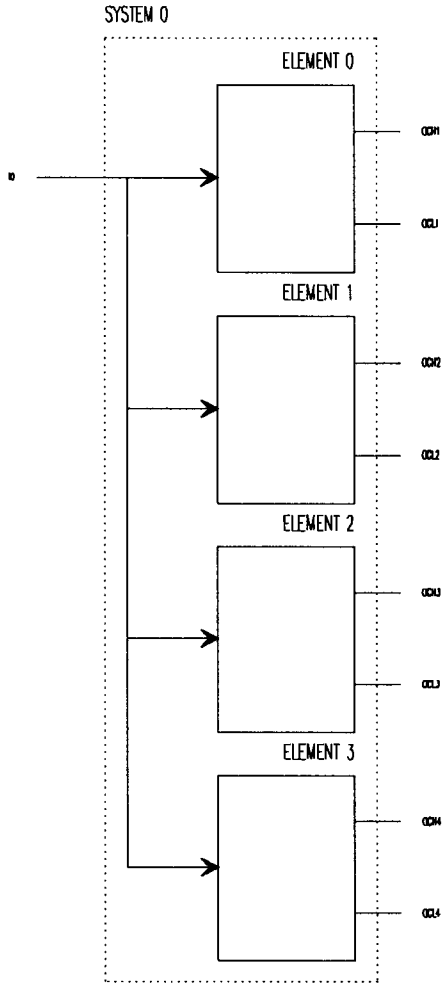
and dc bits. The comparator logic is described by equation (23) or in table form by Table VII. Where qn is the picture bit. A match between the reference bit and the picture bit gives a high output whereas a mis-match gives a low output. If the don't care bit is high the output from the cell is always low (no match).

The results from each comparator cell are added together to give a sum output which is also compared to user defined threshold values. This gives 2 clipped single bit output results labelled OCHx and OCLx in Figure 41.

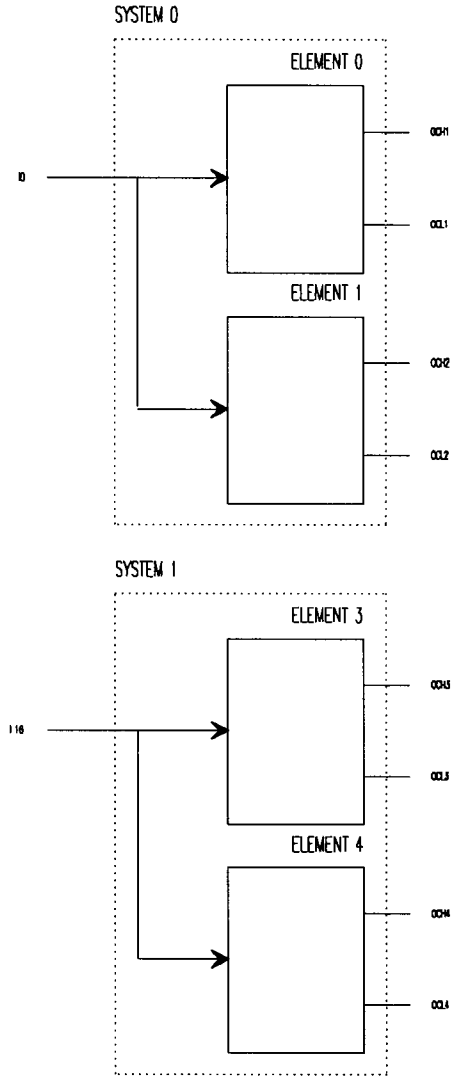
In our case we use 16*16 binary templates in the second layer of the SFMU. By considering the mode tables of the data sheet we have two options, first use mode 20 (1 system containing 4 elements) or second use mode 44 (2 systems containing 2 elements each), see Figure 43. Considering the time needed to reprogram the filters (2600 clock cycles, maximum at full 16MHz) will not cause any problems. So instead of using one IRIS for every subgrid, we use 1 IRIS to serve 2 subgrids. So mode 44 will be used. As a result of this choice we are not able to use the internally provided line delay, because the are to small for our application of compressed line lengths of 2k bit.

**Loading the 16*16 templates into the IRIS**

The memory map of IRIS is shown in Table IX. The "black/white" templates used in mode 44 (2 systems containing each 2 16*16 templates) will be mapped on this memory as listed in Table VIII




MODE 20




MODE 44

**Figure 43**: usable IRIS configurations

The other configuration data and thresholds are loaded to fixed addresses in the IRIS memory (see preliminary data sheet of IRIS)

**Table VIII**: map of template to IRIS memory

| template | arrays | system |
|---|---|---|
| 0 | 1,2 | 0 |
| 1 | 3,4 | 0 |
| 2 | 5,6 | 1 |
| 3 | 7,8 | 1 |

**Table IX**: IRIS memory map

```
array 8              array 4
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx  <- MSB
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx  <- LSB
^112              ^48               <- address
                ^127              ^63
array 7              array 3
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
^96               ^32
                ^111              ^47
array 6              array 2
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
^80               ^16
                ^95               ^31
array 5              array 1
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx 7,0
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx 0,0
^64               ^0
                ^79               ^15
```

This map is only for "white/black" template.

For the "don't care" template add 128 to the address

**Stored Template Ram** for IRIS

The Stored Template Ram of the second layer serves the same goal as the Stored Template Ram of the first layer. Here the IRIS template codes are stored. The organisation and connection of this memory is similar to the one in the first layer. The size of this memory is only 8 bytes, 8 IRIS templates can be stored on the filters. We can not use one memory for first and second layer because for output purposes it might be necessary to address one feature template and one shape template at the same time.

### Output Logic

The Output Logic is a large LUT that needs to be filled by the user before the board becomes operational. A block diagram of this part of the circuitry is given in Figure 44

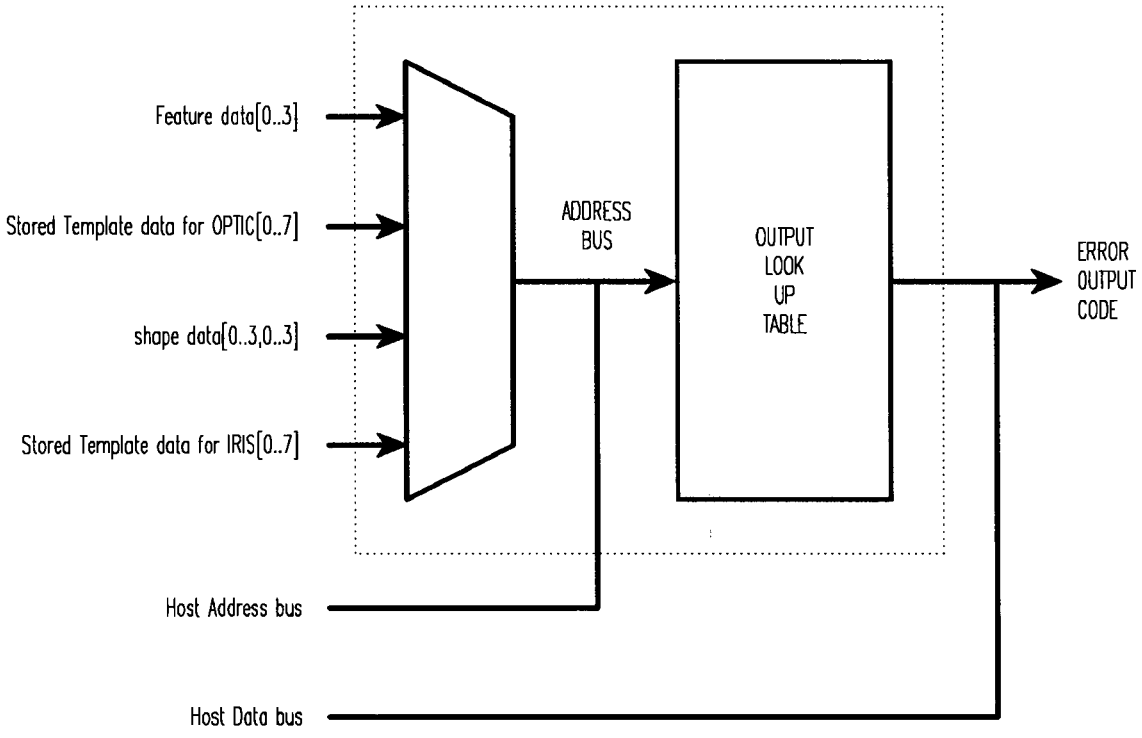



**Figure 44**: block diagram of Output Logic

As the other LUT's used in this architecture, this LUT will also be mapped on the host computer's memory. From Figure 44 follows that the address bus would be 4 + 8 + 16 + 8 = 36 bits corresponding to an address space of 64 Giga byte which is obviously much to big. The data size is reduced by using only the feature data[0..3], only 2 bits per IRIS system (only OCLx are necessary) and since only one filter output will be valid at the same time (subgrids) the source (which filter) can be coded in a 4 bit value (only 12 filter outputs). So when using a CY7C185 (4k * 8) the address will look like Figure 45.

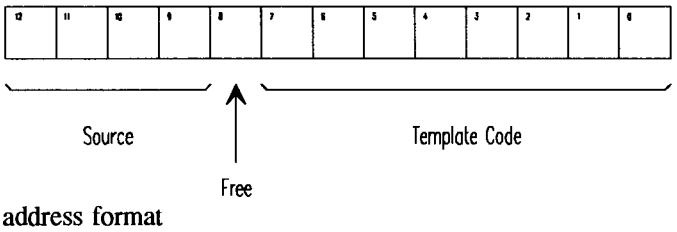



**Figure 45**: output LUT address format

**Conclusions**

At the time of this report I can not draw overall conclusions because the SFMU is not build yet. This is mainly caused by the late arrival of the OPTIC filter (first week of july 1991) but most of the most of the sub-circuits described in the appendix are realized and functioning.

As mentioned in the chapter **Test system setup** I had only one DRC-filter available for simulation.

From the testprogram described in that chapter I can express the strong feeling I have about the proper functioning of the SFMU once it is build and tested.

So at this point two thing remain to be done:

1. Build an test the SFMU
2. Test the CAD generation algorithm

**References:**


[1] Digital Image Processing
Rafael C. Gonzalez / Paul Wintz
second edition
Addison-Wesley Publishing Company 1987

[2] Esprit 2017, Automated Process and Assembly Inspection by 3D Vision
Technical report TR3, 1990

[3] Dynamic Thresholding of Grey-level images
ir. J.A.C. Bernsen
Philips Nat.Lab. report no. 6144

[4] The SBIP DRC-module User Guide
ing. E.F.J. Claessens
Philips CFT, september 1989

[5] L64220 Rank-Value Filter
LSI-logic, 1987

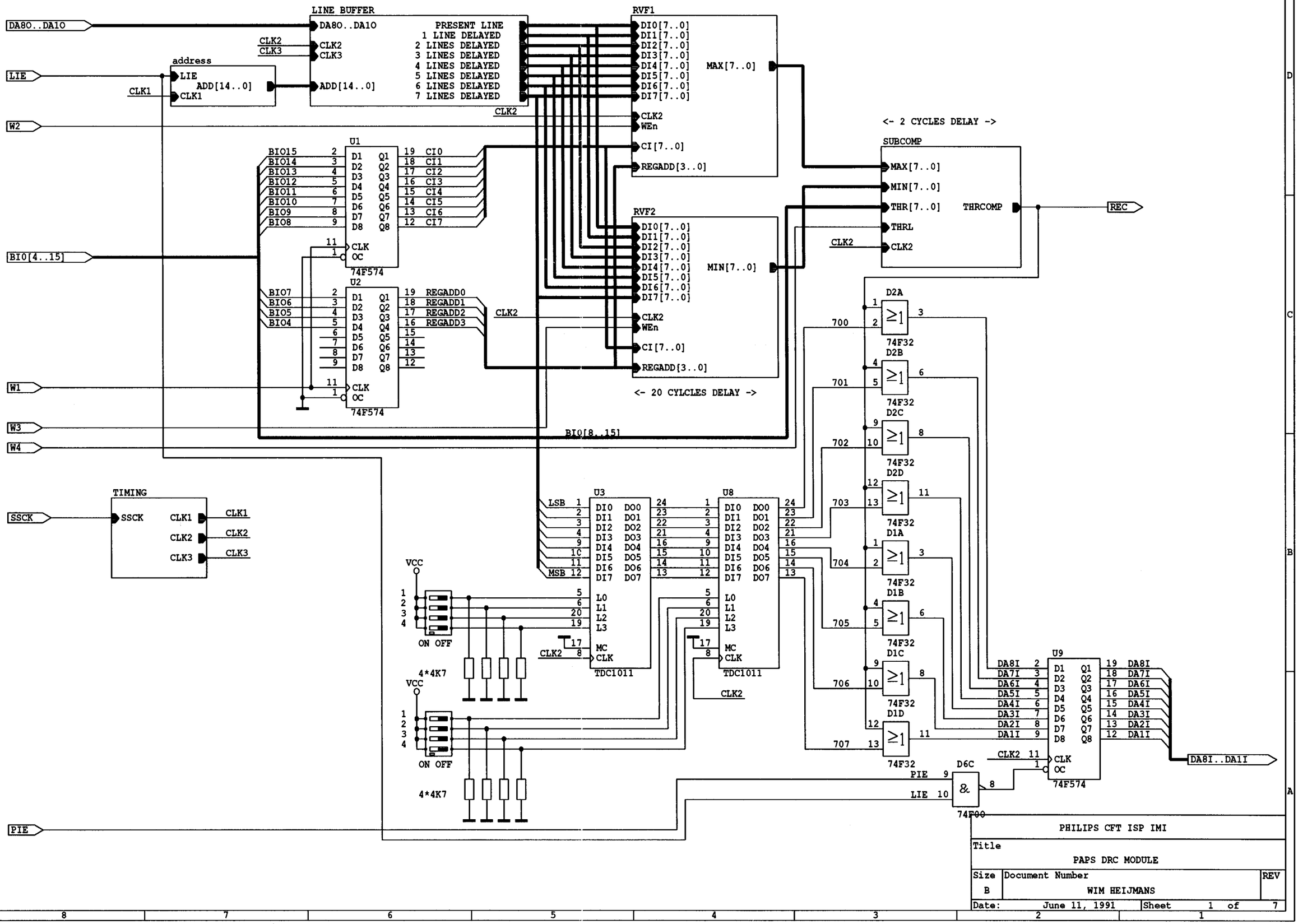
[6] A pipelined MAX-MIN filter image processor
doc.no. 88i06.n.1
S3 Silicon & Software Systems Ltd., 28-9-90

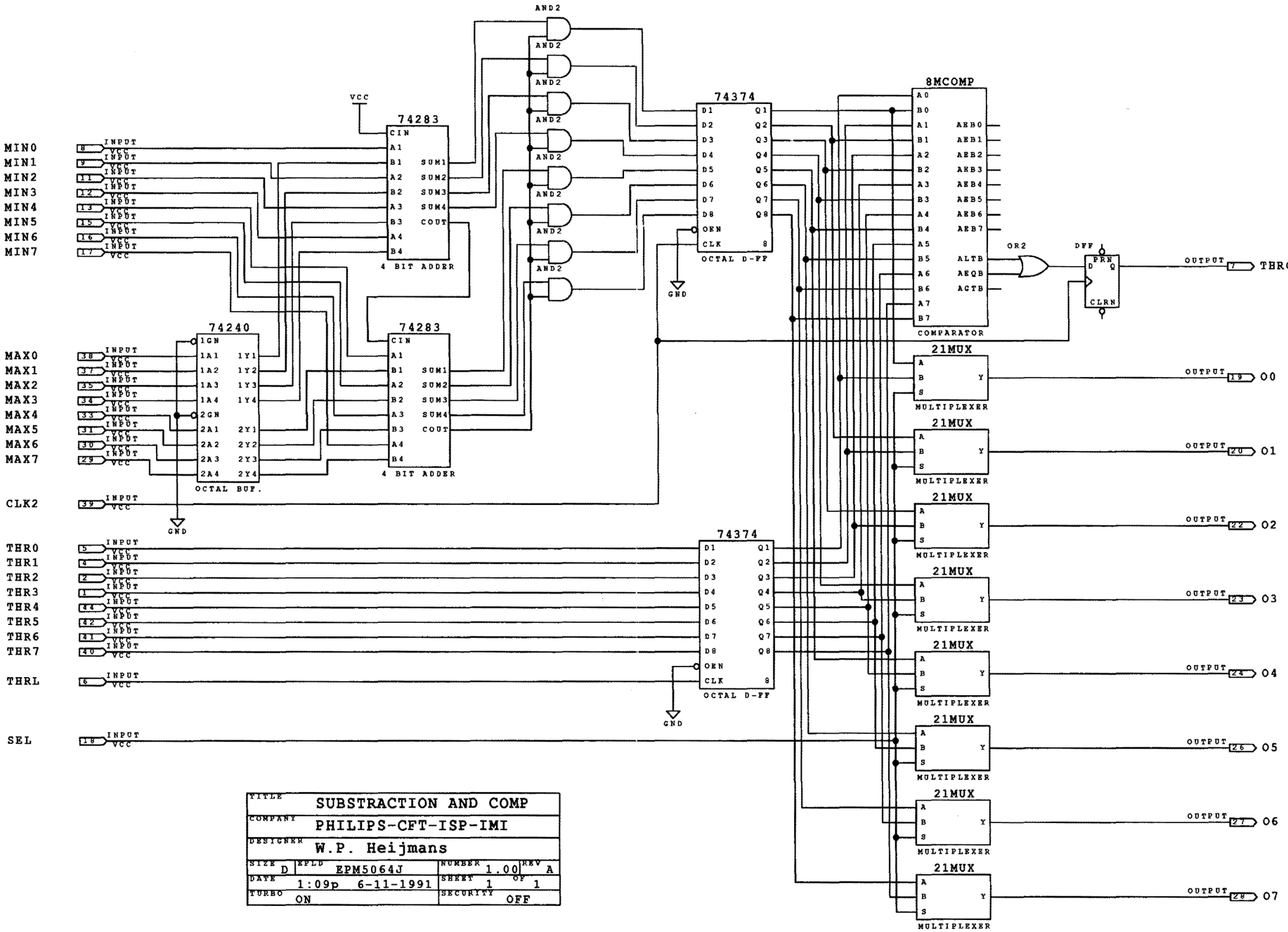
[7] Using the OPTIC Image Processor
doc.no. 88i06.n.2
S3 Silicon & Software Systems Ltd., 11-6-91

[8] IRIS Image Recognition Integrated System
doc.no. 88i04.m.2
S3 Silicon & Software Systems Ltd., 14-3-89

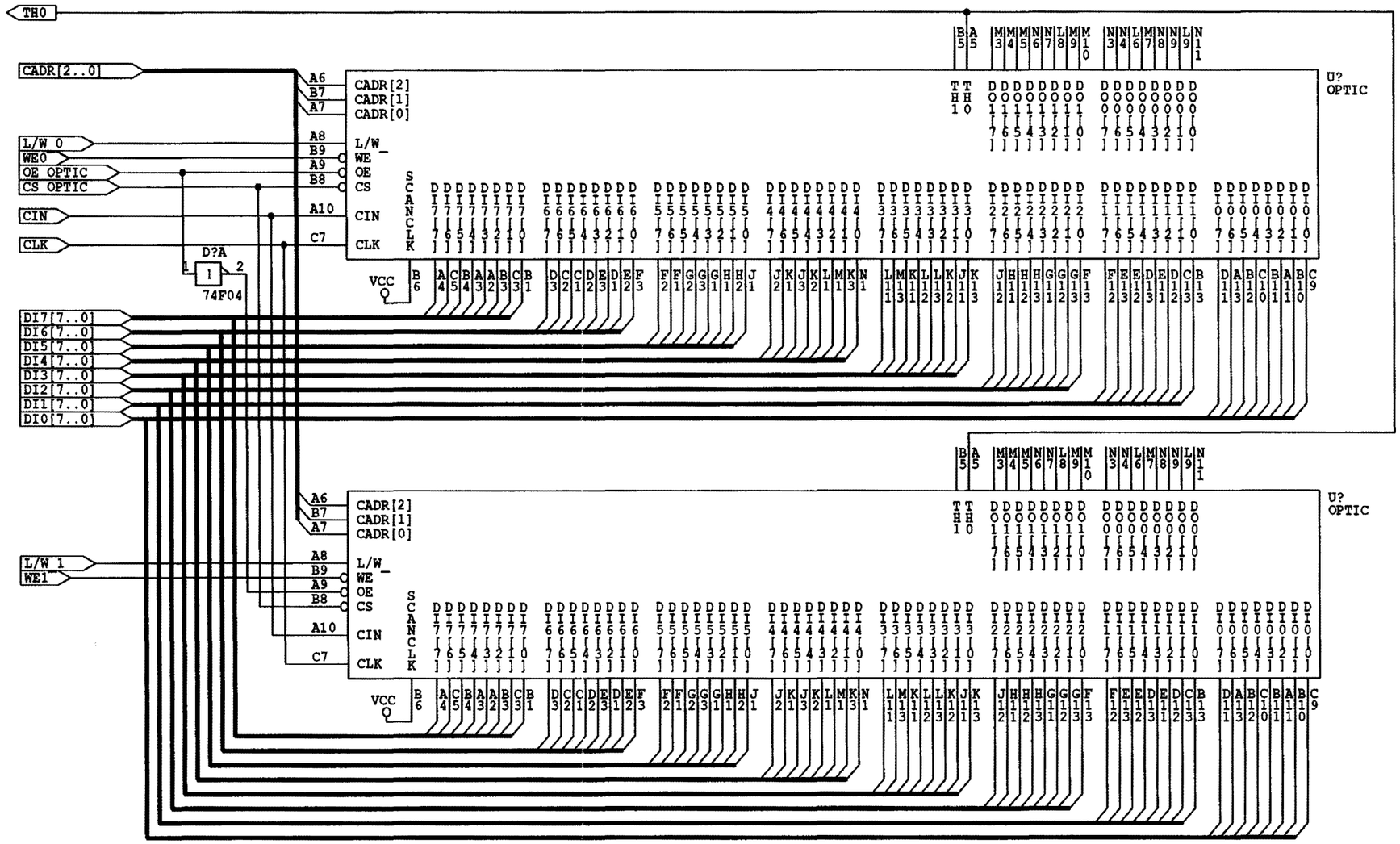
[9] Single Board Image Processor SBIP
Philips CFT
report 17/89EN

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

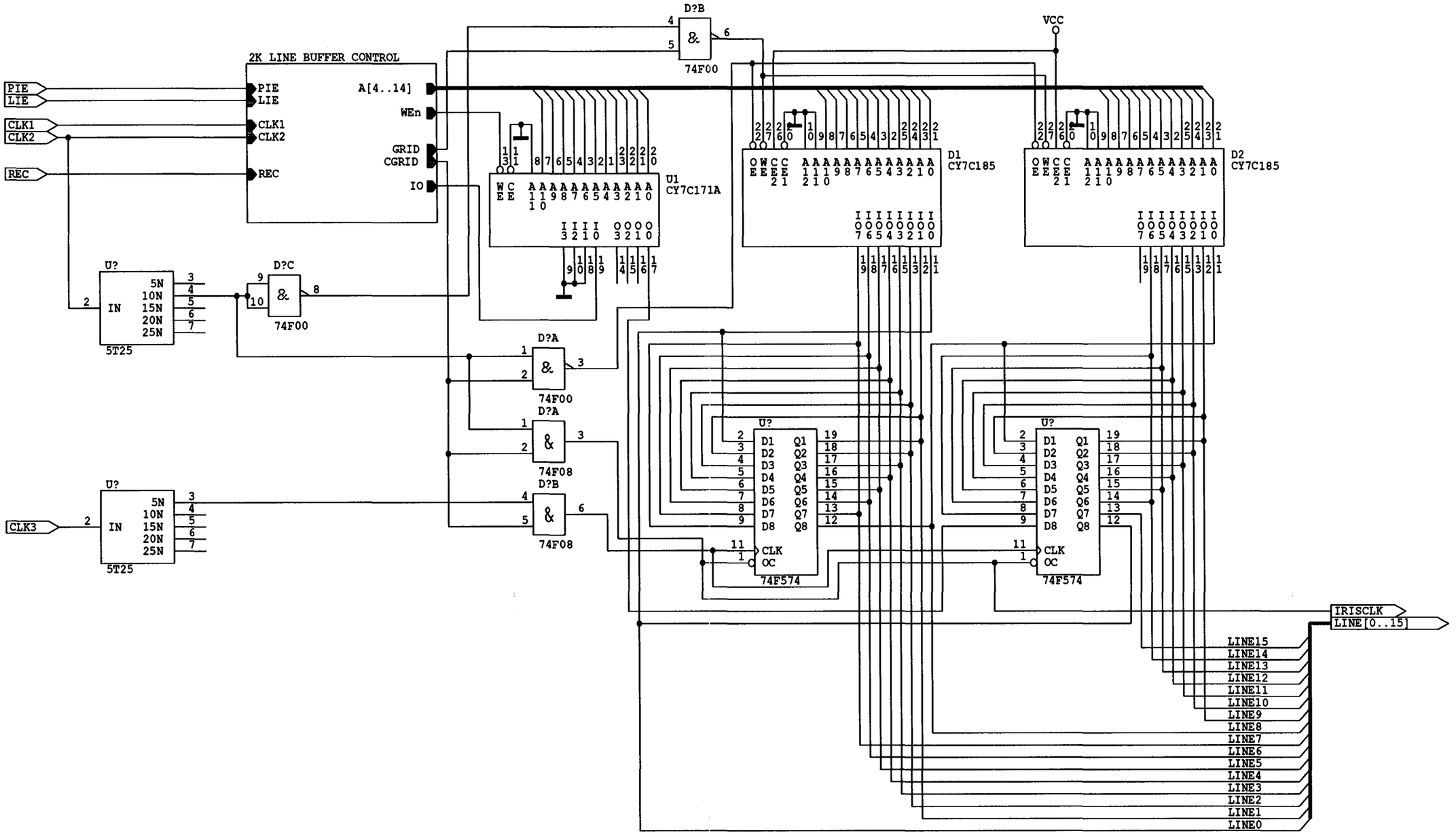

LINE BUFFER
DA80..DA10
CLK2
CLK3
address
LIE
CLK1
ADD[14..0]
PRESENT LINE
1 LINE DELAYED
2 LINES DELAYED
3 LINES DELAYED
4 LINES DELAYED
5 LINES DELAYED
6 LINES DELAYED
7 LINES DELAYED
RVF1
DI0[7..0]
DI1[7..0]
DI2[7..0]
DI3[7..0]
DI4[7..0]
DI5[7..0]
DI6[7..0]
DI7[7..0]
MAX[7..0]
WEn
CI[7..0]
REGADD[3..0]
W2
U1
BIO15
BIO14
BIO13
BIO12
BIO11
BIO10
BIO9
BIO8
D1 Q1
D2 Q2
D3 Q3
D4 Q4
D5 Q5
D6 Q6
D7 Q7
D8 Q8
CI0
CI1
CI2
CI3
CI4
CI5
CI6
CI7
CLK
OC
74F574
BIO[4..15]
U2
BIO7
BIO6
BIO5
BIO4
REGADD0
REGADD1
REGADD2
REGADD3
RVF2
MIN[7..0]
<- 20 CYLCLES DELAY ->
<- 2 CYCLES DELAY ->
SUBCOMP
THR[7..0]
THRCOMP
THRL
REC
W1
W3
W4
BI0[8..15]
TIMING
SSCK
CLK1
CLK2
CLK3
U3
U8
LSB
MSB
DI0 DO0
DI1 DO1
DI2 DO2
DI3 DO3
DI4 DO4
DI5 DO5
DI6 DO6
DI7 DO7
L0
L1
L2
L3
MC
TDC1011
VCC
ON OFF
4*4K7
D2A
D2B
D2C
D2D
D1A
D1B
D1C
D1D
700
701
702
703
704
705
706
707
74F32
U9
DA8I
DA7I
DA6I
DA5I
DA4I
DA3I
DA2I
DA1I
DA8I..DA1I
PIE
LIE
D6C
74F00
PHILIPS CFT ISP IMI
PAPS DRC MODULE
WIM HEIJMANS
Size B
Document Number
REV
Date: June 11, 1991
Sheet 1 of 7



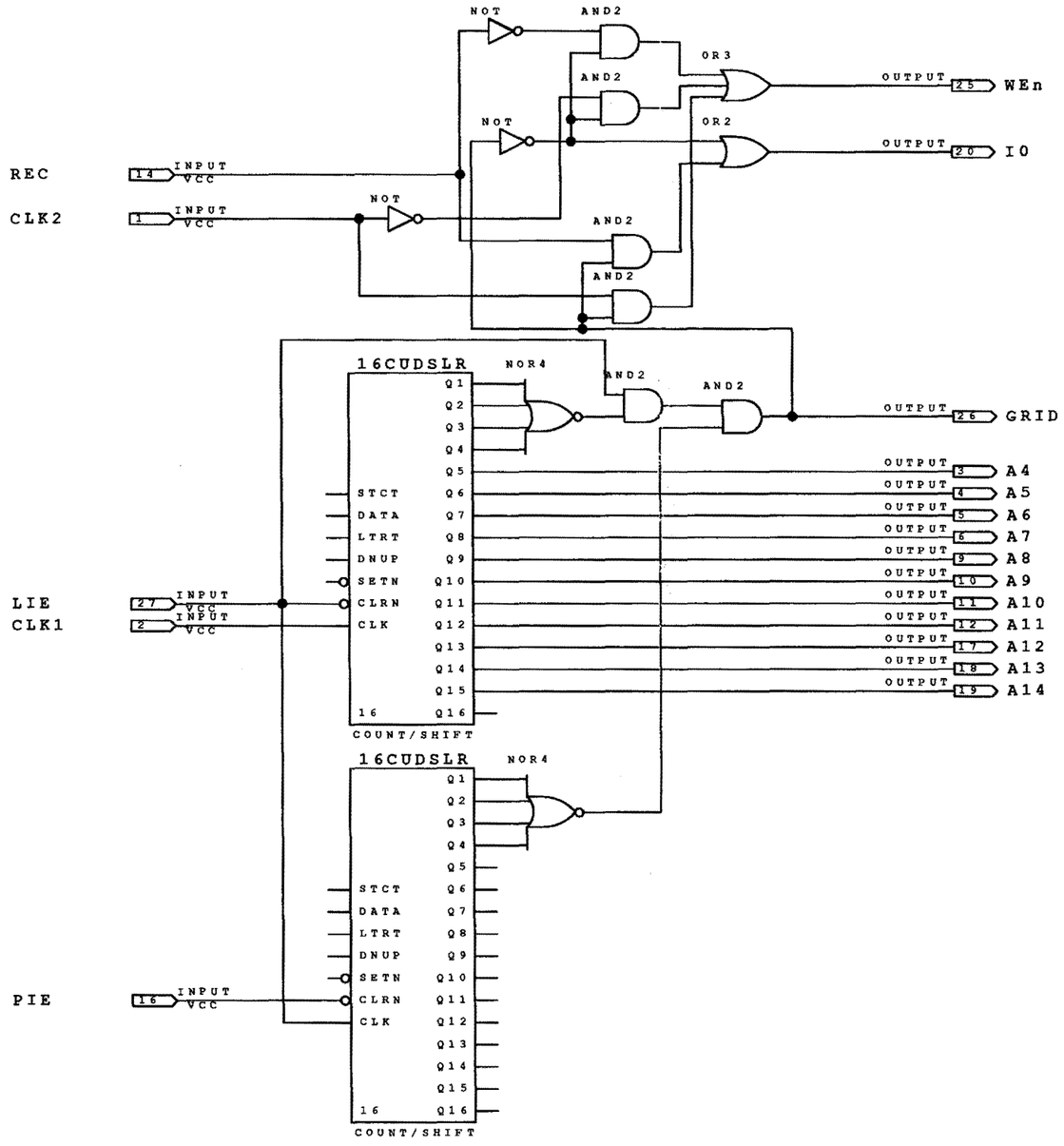

| | | | |
|---|---|---|---|
| TITLE | SUBSTRACTION AND COMP | | |
| COMPANY | PHILIPS-CFT-ISP-IMI | | |
| DESIGNER | W.P. Heijmans | | |
| SIZE D | EPLD EPM5064J | NUMBER 1.00 | REV A |
| DATE 1:09p 6-11-1991 | | SHEET 1 OF 1 | |
| TURBO ON | | SECURITY OFF | |

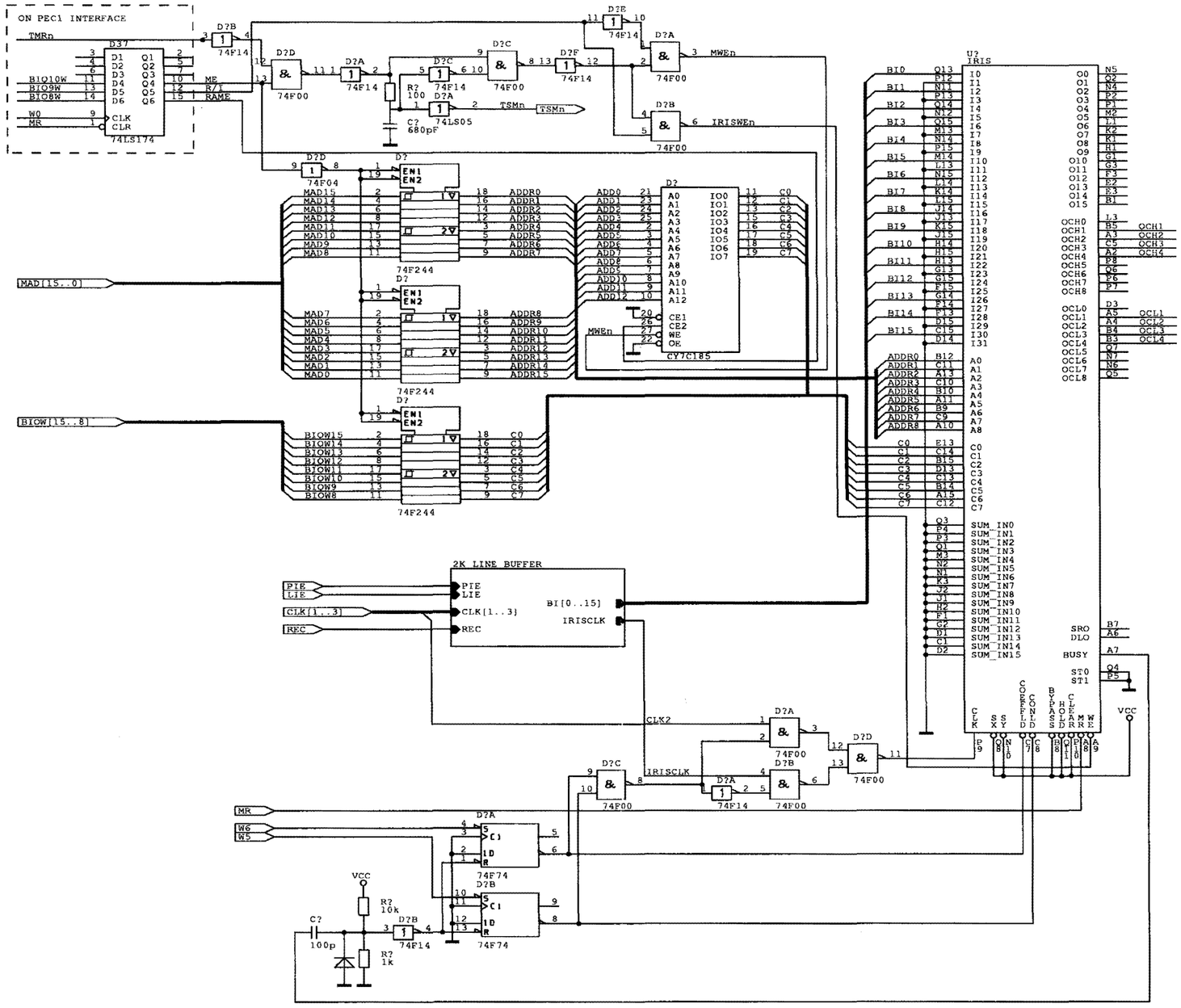



| | | | |
|---|---|---|---|
| PHILIPS CFT ISP IMI | | | |
| Title: FIRST LAYER FILTERS | | | |
| Size: B | Document Number: WIM HEIJMANS | | REV |
| Date: June 18, 1991 | Sheet | of | |




| | | |
|---|---|---|
| PHILIPS CFT-ISP-IMI | | |
| Title: 16 2K LINES LINE BUFFER | | |
| Size: B | Document Number: WIM HEIJMANS | REV |
| Date: May 23, 1991 | Sheet 1 of 1 | |


NOT
AND2
OR3
AND2
OUTPUT 25 WEn
NOT
OR2
OUTPUT 20 IO
REC
14 INPUT VCC
CLK2
1 INPUT VCC
NOT
AND2
AND2
16CUDSLR
NOR4
AND2
AND2
OUTPUT 26 GRID
Q1
Q2
Q3
Q4
Q5
STCT Q6
DATA Q7
LTRT Q8
DNUP Q9
SETN Q10
CLRN Q11
CLK Q12
Q13
Q14
Q15
16 Q16
COUNT/SHIFT
OUTPUT 3 A4
OUTPUT 4 A5
OUTPUT 5 A6
OUTPUT 6 A7
OUTPUT 9 A8
OUTPUT 10 A9
OUTPUT 11 A10
OUTPUT 12 A11
OUTPUT 17 A12
OUTPUT 18 A13
OUTPUT 19 A14
LIE
27 INPUT VCC
CLK1
2 INPUT VCC
16CUDSLR
NOR4
PIE
16 INPUT VCC
COUNT/SHIFT

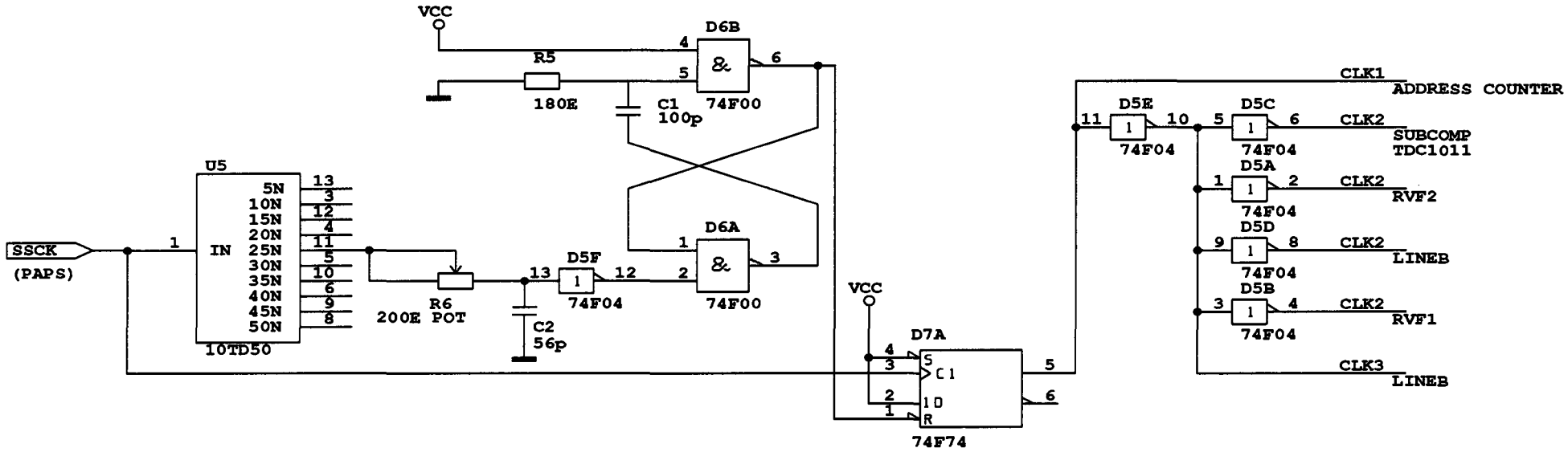



| | | |
|---|---|---|
| PHILIPS CFT ISP IMI | | |
| Title: MEMORY MAPPED MEMORY | | |
| Size: C | Document Number | REV |
| | WIM HEIJMANS | |
| Date: May 23, 1991 | Sheet of | |

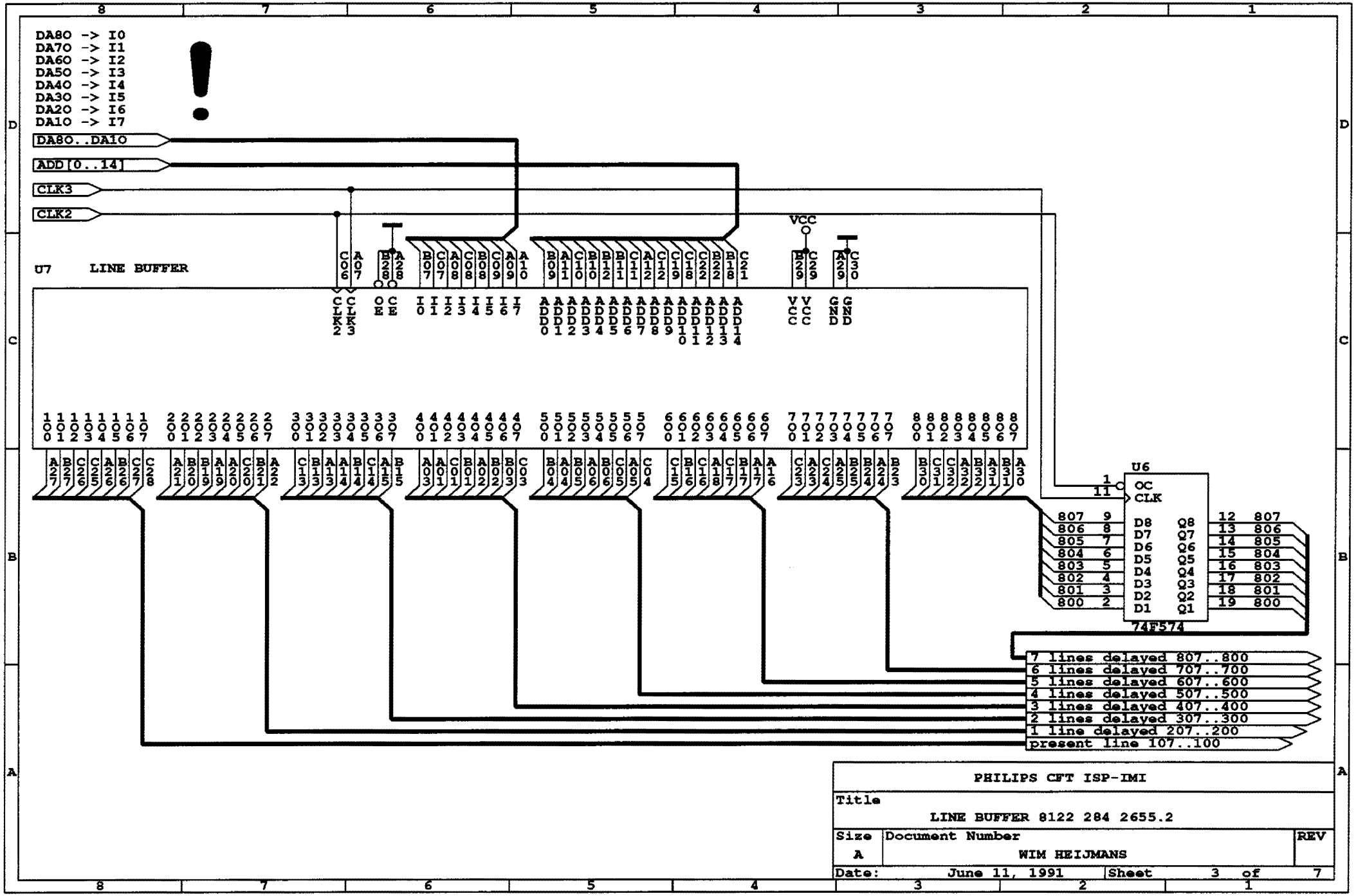
**Appendix:**

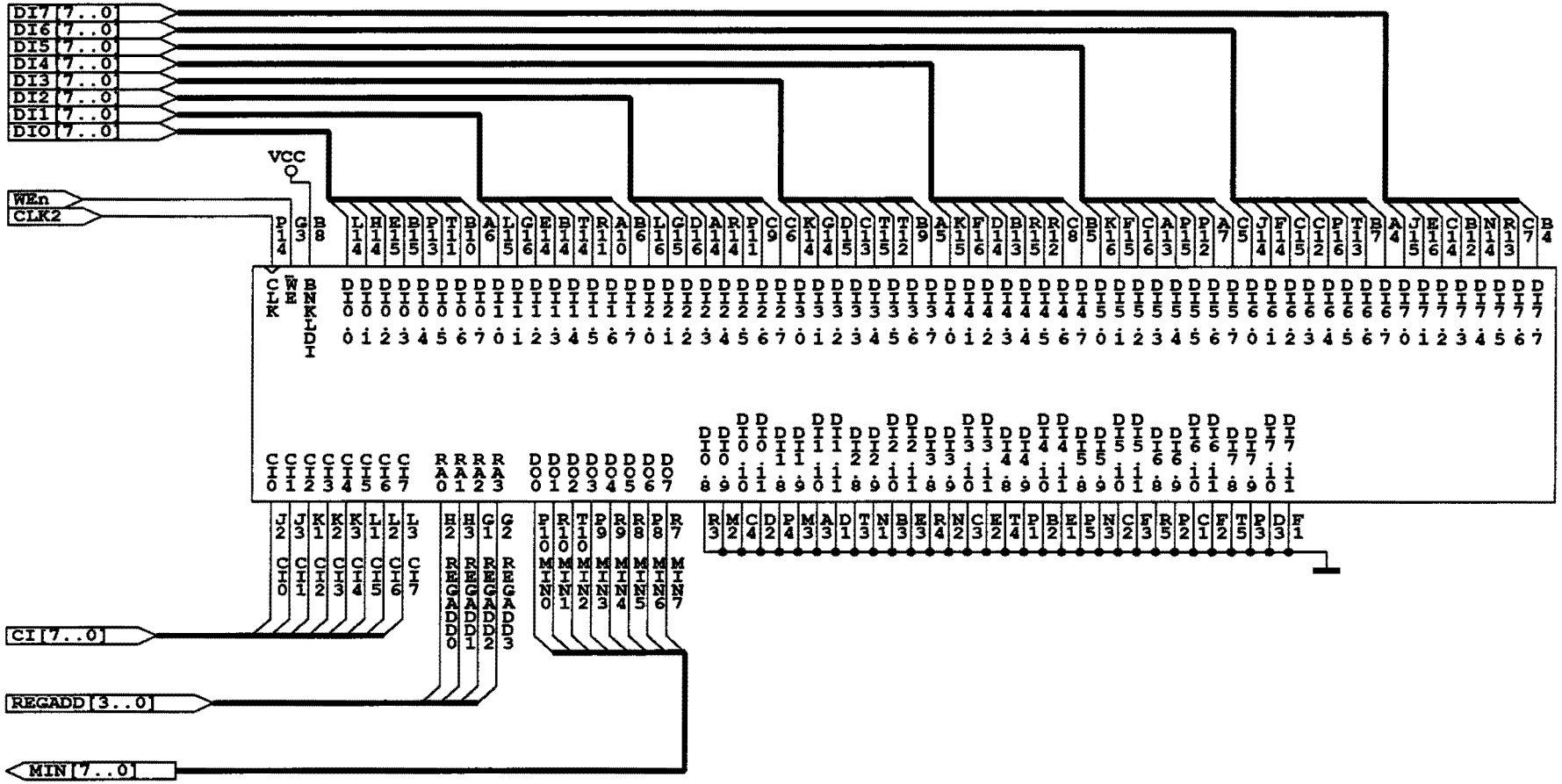
DRC module: 7 sheets

| | |
|---|---|
| PAPS DRC MODULE | 1 of 7 |
| LINE BUFFER TIMING | 2 of 7 |
| LINE BUFFER 8122 284 2655.2 | 3 of 7 |
| RVF1 for MIN | 4 of 7 |
| RVF2 for MAX | 5 of 7 |
| ADDRESS GENERATOR | 2 of 1 |
| SUBSTRACTION AND COMP | 1 of 1 |

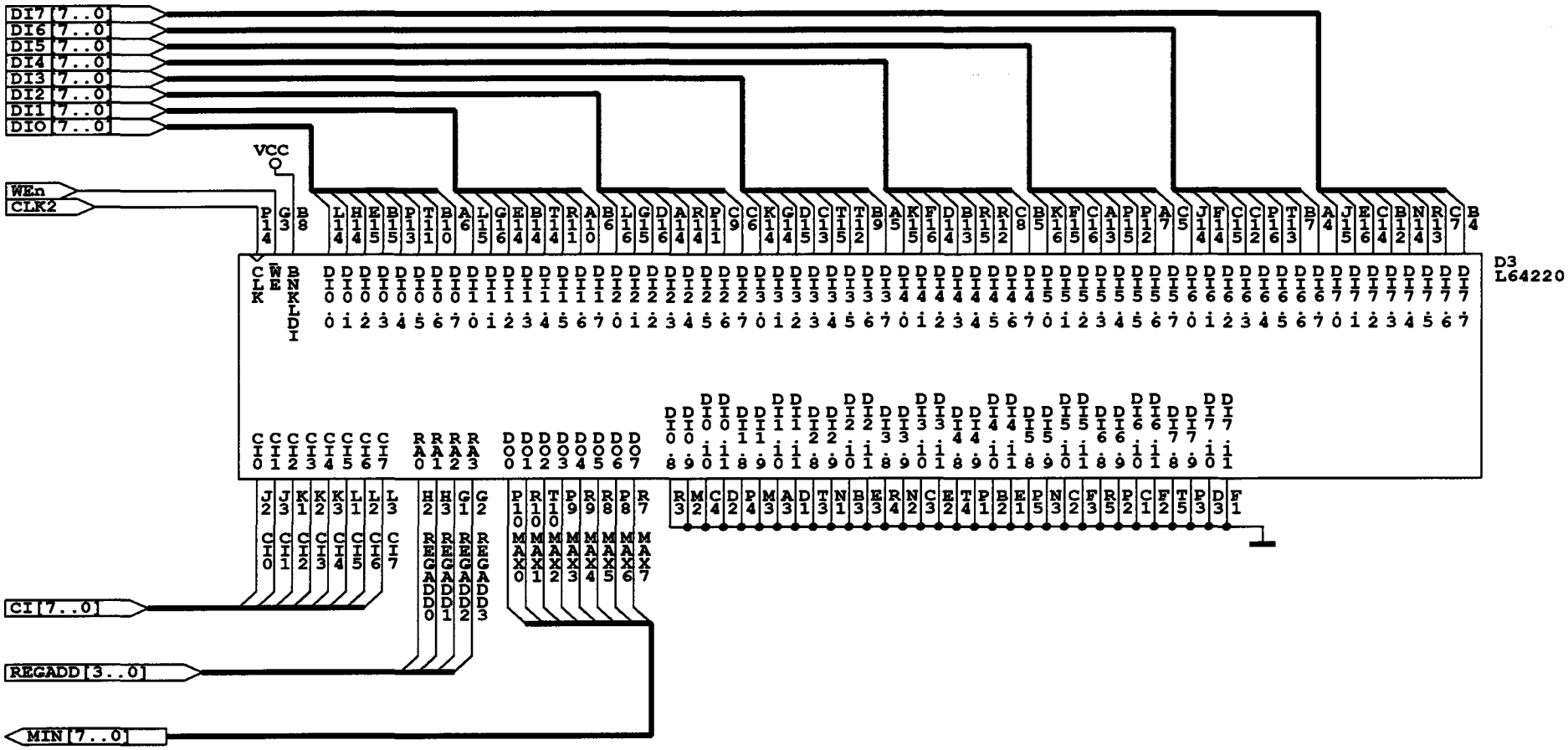



| | | |
|---|---|---|
| PHILIPS CFT ISP IMI | | |
| Title: LINE BUFFER TIMING | | |
| Size: A | Document Number: WIM HEIJMANS | REV: |
| Date: June 11, 1991 | Sheet 2 of 7 | |

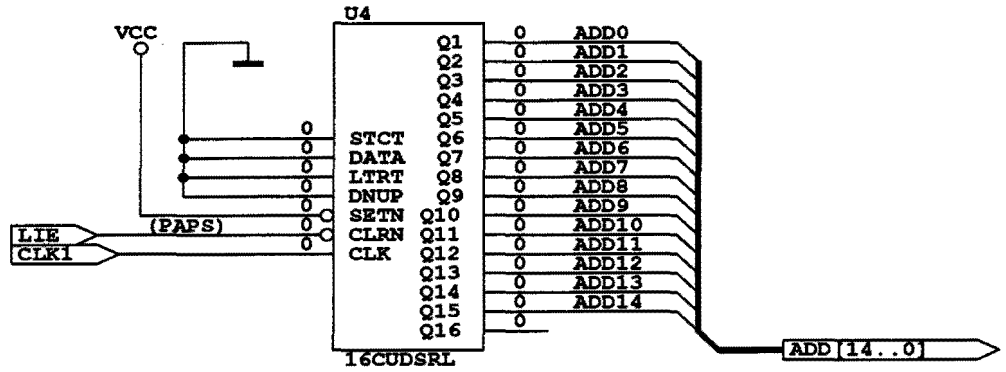

DA80 -> I0
DA70 -> I1
DA60 -> I2
DA50 -> I3
DA40 -> I4
DA30 -> I5
DA20 -> I6
DA10 -> I7
DA80..DA10
ADD[0..14]
CLK3
CLK2
U7 LINE BUFFER
VCC
U6
OC
CLK
D8 D7 D6 D5 D4 D3 D2 D1
Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1
74F574
7 lines delayed 807..800
6 lines delayed 707..700
5 lines delayed 607..600
4 lines delayed 507..500
3 lines delayed 407..400
2 lines delayed 307..300
1 line delayed 207..200
present line 107..100
PHILIPS CFT ISP-IMI
Title
LINE BUFFER 8122 284 2655.2
Size A
Document Number
WIM HEIJMANS
REV
Date: June 11, 1991
Sheet 3 of 7




| | | |
|---|---|---|
| PHILIPS CFT ISP IMI | | |
| Title: RVF1 for MIN | | |
| Size: A | Document Number: WIM HEIJMANS | REV |
| Date: June 11, 1991 | Sheet 4 of 7 | |




| | | |
|---|---|---|
| PHILIPS CFT ISP IMI | | |
| Title: RVF2 for MAX | | |
| Size: A | Document Number: WIM HEIJMANS | REV |
| Date: June 11, 1991 | Sheet 5 of 7 | |




# EPM5032-2

| | | |
|---|---|---|
| PHILIPS CFT ISP IMI | | |
| Title: ADDRESS GENERATOR | | |
| Size: A | Document Number: WIM HEIJMANS | REV: |
| Date: June 11, 1991 | Sheet 2 of 1 | |

**Appendix:**

OPTIC: 3 sheets

FIRST LAYER FILTERS
16 2K LINES LINE BUFFER 1 of 1
2K LINE BUFFER CONTROL

**Appendix:**

IRIS: MEMORY MAPPED MEMORY