

**Median Filtering for Target Detection in an Airborne Threat Warning System**

by

Joseph Paul Havlicek

Thesis submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of  
Master of Science  
in  
Electrical Engineering


APPROVED:

---

John C. McKeeman, Chairman

---

Charles W. Bostian

 Joseph G. Tront

July, 1988

Blacksburg, Virginia

# **Median Filtering for Target Detection in an Airborne Threat Warning System**

by

Joseph Paul Havlicek

John C. McKeeman, Chairman

Electrical Engineering

(ABSTRACT)

Detection of point targets and blurred point targets in midwave infrared imagery is difficult because few assumptions can be made concerning the characteristics of the background. In this thesis, real time spatial prefiltering algorithms that facilitate the detection of such targets in an airborne threat warning system are investigated. The objective of prefiltering is to pass target signals unattenuated while rejecting background and noise. The use of unsharp masking with median filter masking operators is recommended. Experiments involving simulated imagery are described, and the performance of median filter unsharp masking is found to be superior to that of the Laplacian filter, the linear point detection filter, and unsharp masking with a mean filter mask.

A primary difficulty in implementing real time median filters is the design of a mechanism for extracting local order statistics from the input. By performing a space-time transformation on a standard selection network, a practical sorting architecture for this purpose is developed. A complete hardware median filter unsharp masking design with a throughput of 25.6 million bits per second is presented and recommended for use in the airborne threat warning system.

# Acknowledgements

First and foremost, I am forever indebted to my parents for their love, understanding, sound advice, and encouragement.

My greatest appreciation is expressed to Dr. John C. McKeeman. He provided constant encouragement and guidance, repeatedly stepped above and beyond the call of duty to ensure my access to every possible resource, and was a true friend as well as a mentor.

Gratitude is also extended to the other committee members. By allowing me to participate in an undergraduate independent study under his direction, Dr. Joseph G. Tront provided my first exposure to technical research. Dr. Charles W. Bostian gave me the opportunity to be a member of one of the finest research teams at Virginia Tech. I am also grateful for his continuing dedication to the teaching mission of the University.

Without the contributions of P. Willmann Remaklus, the prefilter circuit simulation and verification projects could not have succeeded. He spent long hours with me in the design automation laboratory, and also produced most of the thesis artwork.

At the Naval Research Laboratory, longtime friend Kenneth A. Sarkady assigned this project to me, and also did preliminary design work. Dr. Myron R. Pauli provided insight, guidance, technical consultation, and friendship. Dr. Michael P. Satyshur was kind enough to grant me access to his computer generated imagery.

# Table of Contents

<b>1 Introduction</b>	<b>1</b>
1.1 Problem Impetus	3
1.2 Image Elements and Image Formation	4
1.3 Preprocessing Algorithms	8
1.4 Overview	12
<b>2 Median Filter Unsharp Masking</b>	<b>14</b>
2.1 Prefiltering by Unsharp Masking	16
2.2 Median Filtering	20
2.2.1 Definition of Median Filtering	22
2.2.2 Deterministic Properties	26
2.2.3 Statistical Properties	31
2.2.4 Frequency Domain Characterization	34
2.2.5 Threshold Decomposition	36
2.3 Formalization of the Recommended Algorithm	38
2.3.1 Variant I: $3 \times 3$ Median Filter Mask	39
2.3.2 Variant II: Compensated Median Filter Mask	40

2.3.3 Variant III: One-Dimensional Median Filter Mask .....	41
<b>3 Algorithm Evaluation .....</b>	<b>43</b>
3.1 Measurement Criteria .....	43
3.2 A Simple Experiment .....	47
3.2.1 $3 \times 3$ Laplacian Filter .....	48
3.2.2 $3 \times 3$ Point Detection Filter .....	54
3.2.3 Unsharp Masking with Mean Filter Mask .....	56
3.2.4 Recommended Algorithm, Variant I .....	56
3.2.5 Recommended Algorithm, Variant II .....	58
3.2.6 Recommended Algorithm, Variant III .....	61
3.3 Evaluation on Realistic Inputs .....	63
3.3.1 Experimental Procedure .....	64
3.3.2 Experimental Results .....	67
<b>4 Real Time Sorting for Algorithm Variant II .....</b>	<b>70</b>
4.1 Selection Networks .....	71
4.2 Histogram Method .....	73
4.3 Radix Method .....	75
4.4 Selection Network with In-Place Computation .....	78
<b>5 A Complete Real Time Design .....</b>	<b>84</b>
5.1 Design Specification .....	85
5.2 Design Presentation .....	90
5.2.1 Circuit Module SINGLESTEP .....	93
5.2.2 Circuit Module EXIN SYNC .....	93
5.2.3 Circuit Module DELAY LINE .....	95
5.2.4 Circuit Module SRT MTRX .....	97

5.2.5 Circuit Module MICRO CTRL .....	101
<b>6 Conclusions and Recommendations for Further Research .....</b>	<b>108</b>
6.1 Algorithm Development and Evaluation .....	109
6.2 Algorithm Implementation .....	112
<b>Literature Cited .....</b>	<b>113</b>
<b>Appendix A. Schematics .....</b>	<b>117</b>
<b>Vita .....</b>	<b>140</b>

## List of Illustrations

Figure 1. Geometry of target models	7
Figure 2. Unsharp masking block diagram	17
Figure 3. Block diagram of unsharp masking for the ATWS prefilter	18
Figure 4. Simple frames	28
Figure 5. Mean-squared-error in median and mean filtered noisy step	35
Figure 6. Unit edge frame	49
Figure 7. Input frame with $5 \times 5$ unit amplitude square	50
Figure 8. Smooth monotonic background frame	51
Figure 9. Laplacian filter responses	53
Figure 10. Point detection filter responses	55
Figure 11. Responses using a mean filter mask	57
Figure 12. Responses of recommended algorithm variant I	59
Figure 13. Responses of recommended algorithm variant II	60
Figure 14. Responses of recommended algorithm variant III	62
Figure 15. Selection networks	72
Figure 16. Modified radix method	77
Figure 17. Network to extract the median of eight inputs	79
Figure 18. Comparators with memory	82
Figure 19. Sorting architecture for algorithm variant II	83



Figure 20. Simplified ATWS system block diagram ..... 87

Figure 21. System data path timing characteristics for the ATWS ..... 88

Figure 22. Circuit block diagram ..... 92

Figure 23. Computation of prefiltered outputs ..... 99

Figure 24. Microcode sequencer architecture ..... 102

Figure 25. Microinstruction format ..... 104

## List of Tables

Table 1. SCE and Beta for the input of Figure 8(b) . . . . .	52
Table 2. Average SCE data for the experiment of Section 3.3 . . . . .	65
Table 3. Average beta factors for the experiment of Section 3.3 . . . . .	66

# 1 Introduction

An automatic target recognition system can be defined as a machine that is capable of forming images, and detecting and classifying tactical targets within those images. Any such system must include some type of sensor subsystem which converts scene information to a representation suitable for machine processing. For tactical target detection, scene information is generally collected in the form of midwave infrared radiation. A snapshot of the scene is represented by a matrix, the elements of which are called pixels. The term *pixel* is an abbreviation for *picture element*, where a pixel represents the smallest uniquely definable scene element. The value of each pixel is an integer representing the amount of infrared flux incident on a particular area of the sensor at a particular time. The entire matrix is referred to as a *frame*. In this thesis, the subscript notation  $x_{i,j}$  will generally be employed to reference the pixel located in the  $j^{\text{th}}$  column of the  $i^{\text{th}}$  row of the frame  $X$ . In some cases where the subscript notation becomes too difficult to read, the alternate notation  $x(i,j)$  will be used.

Tactical targets are unfriendly military vehicles, manned or unmanned. The automatic target recognition problem is concerned with the specification of machine algorithms to

analyze frames with the objective of correctly identifying tactical targets. The steps by which this is accomplished are preprocessing, detection, segmentation, feature extraction, classification, tracking, and prioritization [2,4]. In preprocessing, target detectability is improved by enhancing the contrast in frames: target information is amplified or passed, while background and noise information is attenuated or rejected. Detection is the process of identifying those parts of a frame that may contain tactical targets [2,4]. After detection, candidate targets are removed from their surrounding background in a process known as image segmentation [4].

Taken together, preprocessing, detection, and segmentation constitute low level processing in which scene information is represented by pixel magnitudes organized as frames. This representation does not always facilitate target classification, and hence the process of feature extraction is used to map the segmented targets into a more abstract space where representation schemes based upon texture or features may be utilized [2]. Whatever representation is used in higher level processing, classification is the process by which potential targets are identified as specific tactical objects. Determining and monitoring the position of classified targets with relation to the position of the sensor is referred to as tracking, and based on tracking information targets can be prioritized. Once targets are prioritized, specific actions in response to their presence can be initiated.

## ***1.1 Problem Impetus***

Sophisticated automatic target recognition systems with excellent performance have been reported. Bhanu, Politopoulos, and Parvin developed a system for recognizing tanks, trucks, and other military vehicles [3]. In a data base of infrared scenes for which truth information was known, the system detected 85 percent of all targets and correctly classified 80 percent of the detected targets. Running on a Vax 11/780, the system required approximately two minutes of execution time per frame. With the advent of VLSI and VHSIC technologies, as well as recent advances in infrared sensor technology, much research is being devoted to the development of real time automatic target recognition systems [2,3,4,9].

One such system under development at the United States Naval Research Laboratory is an airborne threat warning system for high performance aircraft [9]. The system incorporates a wide-field-of-view staring mode infrared focal plane array sensor. The frame size is 128 by 128 pixels, and the data rate is 120 frames per second. With the sensor mounted on an aircraft, the system recognizes potentially threatening air-to-air and ground-to-air missiles. Henceforth, these missiles will be referred to simply as *targets*. In the scenes observed by the airborne threat warning system, and other similar systems as well, the best known instantaneous characterization of targets is as bright point sources [5,10,11,12]. For investigative purposes, targets are usually simulated by introducing extra impulsive energy into a given background scene by increasing the amplitudes of pixels at the location of the simulated target. The amount by which the amplitudes are increased is generally a function of the scene standard deviation.

This thesis investigates real time preprocessing for the airborne threat warning system, which will hereafter be referred to as the ATWS. The preprocessing must ultimately operate on frames which are representations of scenes observed by the system. Section 1.2 provides a discussion of the elements that make up the scenes and the image formation process by which scene information is assimilated into frames.

## ***1.2 Image Elements and Image Formation***

A frame of infrared imagery comprises target information, background information, and noise. Imaging infrared focal plane array sensors such as the one used in the ATWS are charge integrating devices. Typically, an array of mercury cadmium telluride detectors is used to convert incident infrared radiation into electrical charge. The charge is transferred to an underlying array of silicon storage devices. Charge packets collected in the storage devices are read out sequentially, amplified, and converted to digital data [9]. Among other factors, focal plane array nonlinearity, nonuniformity, and crosstalk will result in the corruption of scene information by noise regardless of the scene under observation [4,5,9,10]. The characterization of this noise is currently an active research area [9,10]. In the ATWS sensor subsystem, signal processing has been implemented to partially correct for sensor nonlinearity and nonuniformity. The effects of focal plane array noise have been modelled in the computer generated imagery used for the experiments of Chapter Three. The imagery was kindly provided by Dr. M.P. Satyshur of the Naval Research Laboratory.

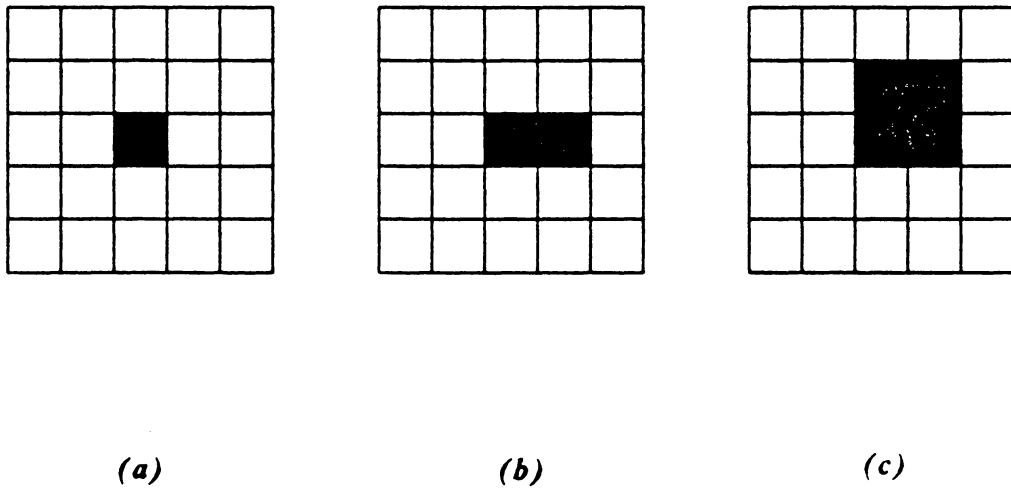
Background refers to the information in a scene that does not arise from a target. Takken, *et al.*, have suggested that target detection is particularly difficult when highly structured background is present [7]. Background and noise features that impede target detection are generally referred to as *clutter*. No unified quantitative definition of clutter exists. Bhanu has defined it as something that looks like a target but is not a target [4]. Hetzler, *et al.*, prefer a broader definition of clutter which embodies all attributes of the background that make the detection of targets more difficult [6]. Their study of clutter in infrared backgrounds suggests that the level of clutter in a scene is strongly related to the global standard deviation of pixel values within the scene. An important assumption made by Takken, *et al.*, and adopted in this thesis is that the spatial extent of the clutter in a scene is strictly greater than the spatial extent of the targets [7]. This assumption involves no loss of generality, since targets are characterized as point sources. The consideration of clutter in the ATWS is extremely important; clutter, rather than image formation noise, has historically been the performance limiting factor in point target detection systems [1,2].

Optical blur is a factor in the image formation process which arises as a consequence of the ATWS sensor's fill factor. The *fill factor* of a focal plane array sensor is the percentage of the array surface area that is covered by active detectors: an array with unity fill factor has no dead spots. All practical imaging array sensors have fill factors less than unity. Inactive bands on the array typically have widths as great as 20 percent of the detector-to-detector spacing [11]. Were the fill factor unity, the location of a target image on the focal plane would not be an issue. Since the fill factor is less than one, a danger exists that targets may be lost in the inactive regions between detectors. To ensure that targets are not lost, the scene must be blurred before imaging. In the ATWS sensor subsystem, the size of the optical blur is the same as the detector spacing

[10,11]. Due to the optical blur, a target centered on a given detector may give rise to excess input energy on adjacent detectors as well. Pauli, Longmire, and Takken [11] have studied the implications of optical blur for automatic target detection and concluded that no single optimal matched linear filter can be defined for the detection of blurred point targets. In this thesis, the optical blurring of targets has been modelled by equally distributing target energy between one, two, or four adjacent pixels. The geometries for the three blurred target models are shown in Figure 1, where each square represents one pixel. The background data used for the experiments of Chapter Three also account for optical blur effects.

To recapitulate the image formation process, frames in the ATWS are quantized representations of the scenes observed by the sensor subsystem. Each frame is an array of integer valued pixels. To a reasonable approximation, the magnitude of each pixel is the sum of a noise term and a term that is linearly related to the amount of infrared flux incident on the corresponding detector in the sensor array. The incident flux arises from background and target information in the scene, which has been perturbed by the optical blur. Highly structured background information and noise may constitute clutter, making the extraction of target information difficult. The objective of preprocessing is to improve target detectability by discriminating against background, noise, and clutter while passing target information.





**Figure 1. Geometry of target models:** (a) 1-pixel target; (b) 2-pixel target smear; (c) 4-pixel target smear.

---

### ***1.3 Preprocessing Algorithms***

A major difficulty in developing preprocessing algorithms, or *prefilters*, for automatic target recognition systems is that few assumptions can be made concerning the character of background, clutter, and targets in infrared imagery. Takken, *et al.*, have described the background characteristics as variable and unknown, and commented that except in the context of specific images, the spatial frequency spectrum of the clutter is also unknown [7]. Hetzler, *et al.*, tried unsuccessfully to formulate quantitative models for backgrounds and clutter [6]. Bhanu has noted a general absence of both analytical scene models and experimental databases [4]. He also noted an absence of target models, and pointed out that recorded signatures are often not repeatable.

Otazo and Parenti have noted that infrared backgrounds are neither stationary nor Gaussian, but have reported some success modelling scenes with first order Markov processes [13]. In their work, matched linear filters were designed for known target signatures. These filters were compared to statistically matched filters designed with *a priori* knowledge of the clutter and target characteristics. They found the two methods to be nearly equal for improving target detectability. Tao, *et al.*, designed nonrecursive Wiener filters and Kalman filters to maximize the target to clutter ratio for extended targets (extended targets are not modelled as point sources and may span several pixels) [12]. They found that the filters performed well against white noise, but were not robust in the presence of varying background clutter. They concluded that the extensive calculations required to estimate background characteristics in real time were not justified.

Due to the unknown and varying characteristics of background and clutter, the general absence of verified target models, and target signature perturbations introduced by optical blur, standard signal detection theory cannot generally be applied to the design of optimum prefilters for point target detection [7,11,13]. Several *ad hoc* methods, as well as linear filters matched for point targets and blurred point targets have been investigated, however. These algorithms are generally categorized as spatial or temporal. Spatial filters map a single input frame into a single output frame by convolving a filter window with the input frame. Given a filter window  $H$ , the general formula for an  $n \times n$  spatial filter ( $n$  odd) with input frame  $X$  and output frame  $Y$  is:

$$y(p,q) = \sum_{i=1}^n \sum_{j=1}^n h(i,j) x(p+i-(n+1)/2, q+j-(n+1)/2) \quad (1.3.1)$$

For a  $3 \times 3$  filter,

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \quad (1.3.2)$$

Various modifications to Equation 1.3.1 have been proposed for use at the edges of frames. For example, ambiguity arises if the equation is applied for the calculation of  $y_{1,1}$ .

The Laplacian and point detection filters have been used extensively for background and clutter suppression because they respond particularly well to point targets [5,7,8,11]. The mask for the Laplacian filter is:

$$\frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad (1.3.3)$$

The point detection filter mask is:

$$\frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (1.3.4)$$

From a practical standpoint, spatial filters are attractive because they do not require the storage of an entire frame. Also, the repetitive and regular nature of linear spatial filtering algorithms makes them particularly well suited to implementation in parallel pipelined hardware.

While spatial filtering algorithms operate on single frames, temporal algorithms make use of the fact that successive frames represent successive snapshots of the scene. If the sensor were not moving, then a simple subtraction of two successive frames would remove all stationary background information. Fast moving targets and random noise would be passed by the differencing operation. The output frame of a temporal differencing filter is a linear combination of the past and present input frames. Employing the euclidean inner product notation used by Bergen and Mazaika [8], the equation for a first order temporal differencing filter is:

$$\begin{aligned} y(t, i, j) &= \left\langle \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} x(t, i, j) \\ x(t-1, i, j) \end{bmatrix} \right\rangle \\ &= x(t, i, j) - x(t-1, i, j) \end{aligned} \quad (1.3.5)$$

One realization of second order temporal differencing is:

$$y(t, i, j) = \left\langle \begin{bmatrix} -.5 \\ 1 \\ -.5 \end{bmatrix}, \begin{bmatrix} x(t, i, j) \\ x(t-1, i, j) \\ x(t-2, i, j) \end{bmatrix} \right\rangle \quad (1.3.6)$$

Applications such as the ATWS require that the sensor be in motion. This results in some clutter *leaking*, or being passed through, a temporal differencing filter. Clutter leakage becomes more pronounced as the order of the temporal differencing filter is increased. Excellent clutter reduction has been achieved through the use of low order temporal differencing filters in conjunction with high frame rate sensors. In a study by Fraedrich, low order temporal differencing was found to reduce clutter standard deviation by factors of 30 to 70 [1]. Also using low order temporal differencing, Pauli, Longmire, and Takken reported a forty-fold clutter reduction [11]. In general, the primary drawback of temporal differencing is that a number of frames equal to the order of the algorithm must be stored by the filter. In real time implementations, this would necessitate the use of large, ultra high speed random access memories and creative addressing schemes.

Spatial filters having clutter reduction capabilities equal to those of temporal filters have not been reported. In fact, Pauli, Longmire, and Takken found that when spatial and temporal algorithms were used together, the performance of the temporal algorithms was frequently degraded. Only in rare cases did the addition of a spatial algorithm improve the performance of a temporal filter [11]. One should bear in mind that the dramatic results cited in the preceding paragraph pertain to the clutter reduction capabilities of temporal differencing algorithms. The ability of these algorithms to discriminate against clutter while passing targets was not addressed.

## *1.4 Overview*

In this thesis, nonlinear spatial algorithms for prefiltering in the ATWS are developed and investigated. The algorithms are collectively called *median filter unsharp masking*. They are based on order statistics, and involve subtraction of running local medians from the input frames. Median filter unsharp masking was suggested to the author by researchers at the U.S. Naval Research Laboratory. During the period of September 1987 to June 1988, the author devoted considerable effort to a theoretical investigation of median filter unsharp masking. Arguments supporting its use for prefiltering in the ATWS were formulated. The investigation is summarized in Chapter Two, where the author also recommends three specific median filter unsharp masking algorithm variants for use in the ATWS. During the period of December 1987 to July 1988, the author conducted computer simulation experiments to quantitatively evaluate median filter unsharp masking for automatic target detection prefiltering applications. Several other algorithms were also experimentally investigated. Results from two experiments are presented in Chapter Three. Design considerations for the implementation of real time median filters are discussed in Chapter Four. While working at the United States Naval Research Laboratory during summer 1987, the author designed a complete real time prefilter based on median filter unsharp masking. Under the supervision of the author and Dr. J.C. McKeeman, a Virginia Tech team validated the design during the fall of 1987. The design is presented in Chapter Five.

The following list summarizes the specific contributions of this thesis:

1. Three median filter unsharp masking prefilter algorithms are evaluated on their ability to enhance detectability of point targets and blurred point targets in typical ATWS imagery. For the scenes that are investigated, two of these algorithms are shown superior to three commonly used linear algorithms.
2. A new algorithm evaluation criterion called the beta factor is proposed in Section 3.1. For certain applications, the beta factor may be more informative than previously proposed algorithm evaluation criteria.
3. A compensated median filter masking operator is introduced. For the scenes that are investigated, the compensation is shown to offer no tangible advantage over an uncompensated median filter.
4. A special hardware sorting architecture for real time median filtering is presented in Section 4.4.
5. Through the presentation of a complete design, median filter unsharp masking is demonstrated to be a practical algorithm for use in real time automatic target detection systems.

Chapter Six is devoted to conclusions and recommendations for further research.

## 2 Median Filter Unsharp Masking

In Section 1.2 the image formation process for the ATWS was discussed. The resulting image model can be summarized as:

$$\text{frame} = \text{targets} + \text{background} + \text{noise} \quad (2.0.1)$$

The objective of prefiltering is to facilitate detection of the targets. In this respect, both the background and noise constitute clutter. The image model can be simplified to:

$$\text{frame} = \text{targets} + \text{clutter} \quad (2.0.2)$$

To improve target detectability, we seek a prefiltering algorithm that will attenuate the clutter while passing the target information. We define the *ideal algorithm* as an imaginary filter whose output is zero for all non-target pixels. The ideal algorithm also passes all targets without attenuation. Unfortunately, the ideal algorithm cannot be realized because it requires *a priori* knowledge of the target locations and amplitudes. In this chapter, we will attempt to define a realizable algorithm which closely approximates the ideal algorithm.



In Section 1.1 we noted that targets in the ATWS are characterized as bright point sources. They are impulsive in nature, and consequently an input frame will contain relatively high spatial frequencies in the neighborhood of a target. The separation of an input signal into a high frequency noise component and a relatively lower frequency information bearing component is a common problem in signal processing. For example, a coded voice signal might be corrupted by high frequency noise during transmission through a channel. Attenuation of noise has been the object of extensive investigation, and a class of filters known as *smoothers* has been established to remove high frequency noise from input signals.

Since targets are characterized by high spatial frequencies, it would seem that an appropriate smoother could remove targets from a frame. If the targets were smoothed away and the smoothed frame were subsequently subtracted from the original input frame, then the clutter would be removed without attenuating the targets. We would then have an approximation to the ideal algorithm. Such subtraction of a modified, or *masked* version of an image from the original is the basis for an image enhancement technique known as *unsharp masking*. In Section 2.1 the use of unsharp masking to approximate the ideal algorithm is proposed, and incorporation of a nonlinear masking operator known as the *median filter* is suggested. The properties of median filters are discussed in Section 2.2. In Section 2.3, three specific median filter unsharp masking algorithms are defined to approximate the ideal algorithm. In some of these discussions, frames are treated as data sequences in a theoretical sense. Hence, the term *sample* is used interchangeably with the term *pixel*.

## 2.1 Prefiltering by Unsharp Masking

The general equation for unsharp masking is:

$$Y = G(1 - \alpha)[X - H(X)] + \alpha X \quad (2.1.1)$$

where

- $X$  = the original input image
- $H(\cdot)$  = the masking operator
- $G(\cdot)$  = an arbitrary gain function
- $\alpha$  = a mixing parameter,  $0 \leq \alpha \leq 1$
- $Y$  = the enhanced output image

An unsharp masking system block diagram is shown in Figure 2. Choice of the smoothing operator  $H(\cdot)$  is extremely important in the application of unsharp masking to any particular problem. Lo has used unsharp masking to compress the global dynamic range of images while increasing local area contrast by using a  $7 \times 7$  linear low-pass filter for  $H(\cdot)$  [14]. His objective was to improve the quality of the images on a CRT display having limited luminance sensitivity. In his study, 21 observers evaluated filtered images on the basis of psychophysical criteria. Of the six algorithms evaluated, unsharp masking was ranked third best.

If a smoother  $H(\cdot)$  can be found to remove targets from a frame without significantly distorting the clutter, then an approximation to the ideal algorithm can be realized by setting  $G(\cdot) \equiv 1$  and  $\alpha \equiv 0$  in Equation 2.1.1. A block diagram of the resulting system is shown in Figure 3. The ATWS prefilter problem is then reduced to that of finding a suitable smoother that is capable of removing targets without distorting the clutter.

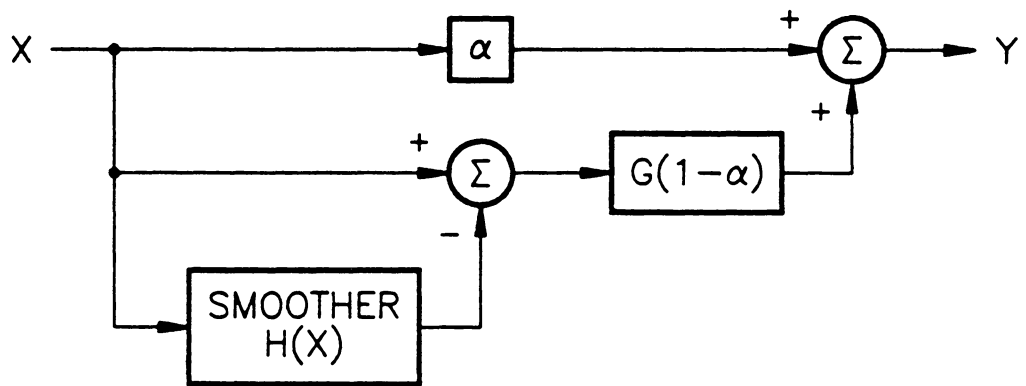
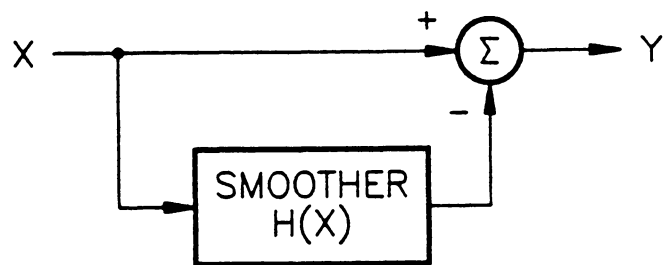


Figure 2. Unsharp masking block diagram

---



**Figure 3. Block diagram of unsharp masking for the ATWS prefilter**

---

In most smoothing applications, the information bearing signal components are spectrally disjoint from the noise, and hence a linear filtering operation can be used to separate them [29]. However, as was noted in Section 1.3, no assumptions can be made regarding the spectral content of the clutter in the ATWS. In particular, sharp edges in the background may contain the same spatial frequencies that are characteristic of targets. As was pointed out by Garibotto and Lambarelli in their study of noise removal in biomedical images, linear filtering is often useless for the removal of noise (or targets in the ATWS) when the signal components are not spectrally disjoint [19]. They noted that the application of linear smoothers resulted in distortion of sharp discontinuities in the data (or clutter in the ATWS). For an unsharp masking algorithm to be effective as a prefilter for the ATWS, the smoother must preserve sharp, discontinuous background features. For example, if a corner were significantly distorted by the smoother, then it might appear as a target upon subtraction of the masked image from the original.

Beaton and Tukey faced a similar problem in fitting polynomials to band-spectroscopic data [18]. They searched for a smoother whose output would not be affected by the presence of localized maxima and minima in the input. The likeness of this problem to the ATWS prefilter is striking. Targets represent local maxima, and we seek a smoother that will pass all clutter features undistorted. For effective subtraction in the unsharp masking algorithm, the ability of the smoother output to locally track clutter features must not be affected by the presence of a target in the input. Beaton and Tukey concluded that the robustness offered by nonlinear smoothing operators was required for their application. In particular, they recommended the use of a median filter. Gray, McCaughey, and Hunt have used unsharp masking with a median filter mask for edge enhancement and noise cleaning of digital images [15]. They compared the performance

of a linear mean filter mask to that of the median filter mask and found that the median filter was superior when the noise contained many sharp edges. Garibotto and Lambarelli described the median filter as a smoother that is capable of removing isolated features by replacing them with the local background [19]. Since that is precisely the capability we desire in a masking operator for the ATWS prefilter, use of a median filter masking operator is strongly recommended by the author. Henceforth, we will refer to Equation 2.1.1 with  $G(.) \equiv 1$  and  $\alpha \equiv 0$  as *the recommended algorithm* when a median filter is used for  $H(.)$ . In Section 2.2, the the properties of median filters are discussed.

## 2.2 Median Filtering

Median filters were first described by Tukey, who initially used them for smoothing economic time series [16,17,18]. He modelled data sequences according to:

$$\text{data} = \text{fit} + \text{residuals} \quad (2.2.1)$$

He referred to the fit as *the smooth*, and the residuals as *the rough*. With respect to the ATWS prefilter, the clutter represents the smooth, while the targets represent the rough. Tukey observed the ability of median filters to remove the rough without distorting sharp features in the smooth, and noted their ability to do this without precise statistical models of either data component. Median filters are members of a general class of nonlinear systems called *order statistic filters*. The output of these filters is at each point a linear combination of the local order statistics of the input. Bovik, Huang, and Munson have studied order statistic filters and proven that the optimum order statistic filter tends toward a median filter as the signal components to be removed become more

impulsive in nature [47]. They also commented that the nonlinearity inherent in the ranking of order statistics causes the analysis and design of such filters to be extremely difficult.

Since their introduction, median filters have been applied primarily to speech processing and image processing. Steele and Goodman used them to smooth out transmission errors in linear PCM [23]. Jayant used them to correct flipped bits in PCM digital speech signals [21]. He found that the median filter was superior to a linear mean filter, provided that the random variables used to model error occurrences were independent. This result seems reasonable since, in a neighborhood of three objects two of which are erroneous, the likelihood is high that the median of the three will itself be in error. Jayant specifically noted the ability of the median filter to correct transmission errors without smearing the speech waveform.

In a widely cited study, Rabiner, Sambur, and Schmidt investigated the removal of local rough from speech signals [20]. They found that the median filter performed well, and that linear smoothers were inadequate because of their tendency to smear the waveforms. Additionally, the combination of a median filter followed by a linear smoother was found to be even more effective than the median filter alone. Bovik, Huang, and Munson have contended that the excellent performance of the combination smoother was a consequence of the specific applications investigated by Rabiner, Sambur, and Schmidt, and that there is no evidence to support the use of such a configuration in general [47].

Pratt was among the first to apply two-dimensional median filters to image processing [24,25]. Proceeding qualitatively, he used them to clean both impulsive and salt-and-pepper noise from digital images while preserving edges. *Salt-and-pepper* refers

to noise which contains both positive and negative impulses. In another early study, Frieden used median filters to remove spurious overshoot and undershoot from digital images after the application of an edge enhancement operator [26]. Median filtering will be precisely defined in the next section.

### 2.2.1 Definition of Median Filtering

Among an odd number of samples, the median is the one with the middle value. For an even number of samples the median is generally defined as the average of the two samples that are middle in value [43]. Suppose that we have a set of  $n$  samples  $X = \{x_i \ni 1 \leq i \leq n\}$ . The sequence  $R = \{r_i \ni 1 \leq i \leq n\}$  of order statistics of  $X$  is defined by

$$R = \text{sort}[X] \quad (2.2.2)$$

where  $\text{sort}[\cdot]$  is an operator whose output samples are the input samples arranged in ascending order by value. For example, in Equation 2.2.2,  $r_1$  is the smallest element of  $X$ , while  $r_n$  is the largest element of  $X$ . If  $n$  is odd, then the median  $v$  of  $X$  is defined by:

$$v = r_k \quad (2.2.3)$$

where

$$k = \frac{(n + 1)}{2}$$

If  $n$  is even, then:



$$v = \frac{r_k + r_{k+1}}{2} \quad (2.2.4)$$

where

$$k = \frac{n}{2}$$

To simplify this notation, we define the median operator  $\text{MF}[\cdot]$  and write:

$$v = \text{MF}[X] \quad (2.2.5)$$

An important point is that insofar as the operator  $\text{MF}[X]$  is concerned, the dimensionality of  $X$  is irrelevant: the sequence  $R$  is always one-dimensional. Median filtering involves the computation of running medians on the input. In response to the input sequence  $X = \{x_i\}$  (of arbitrary length), the output  $Y = \{y_i\}$  of a one-dimensional median filter of length  $N$  ( $N$  odd) is defined as:

$$y_i = \text{MF}[W] \quad (2.2.6)$$

where

$$W = \{x(i - (N - 1)/2), \dots, x(i + (N - 1)/2)\}$$

$W$  in Equation 2.2.6 is called the *window* of the filter, and one can imagine the window sliding along the input sequence as successive samples of  $Y$  are computed. The extension of Equation 2.2.6 to even values of  $N$  is straightforward. In this case, however, the window cannot be symmetrically centered about  $x_i$  during the computation of  $y_i$ . The definition of a two-dimensional median filter simply involves the specification of a two-dimensional window. For example,  $y_{i,j}$  might be computed from a  $3 \times 3$  square

window with  $x_{i,j}$  at its center. Normally, reference to an  $M \times M$  median filter implies a square two-dimensional window of size  $M \times M$ , where  $M$  is understood to be odd.

When finite sequences are considered, a problem arises in using Equation 2.2.6 to compute the filter outputs near the ends of a one-dimensional sequence or at the edges of a two-dimensional sequence. A complete neighborhood of input samples does not exist at these points, and hence samples are missing from  $W$ . Several conventions have been proposed for dealing with the problem by appending extra samples to the input sequence. In this thesis, the output at such points will be considered indeterminate and meaningless.

Henceforth, when no danger of ambiguity exists, we shall relax the definition of the symbol  $\text{MF}[\cdot]$  by allowing a sequence to appear on the left of the equals mark, and write  $Y = \text{MF}[X]$  to mean that the sequence  $Y$  results from the application of a median filter to the sequence  $X$ . The sequences  $X$  and  $Y$  will always be assumed to have identical dimensions. The notation  $y_i = \text{MF}[\{x_1, x_2, x_3\}]$ , in which a single sample appears on the left of the equals mark, will still be used to mean that  $y_i$  is the median value of the samples  $\{x_1, x_2, x_3\}$ . As an example of median filtering, consider the finite length step sequence  $X = \{6, 6, 6, 6, 12, 12, 12, 12\}$ . If we apply a one-dimensional median filter of length three to  $X$  and use the symbol  $\emptyset$  to represent an indeterminate value, then the output sequence  $A = \text{MF}[X] = \{\emptyset, 6, 6, 6, 12, 12, 12, \emptyset\}$ . This filter is often referred to as a *three-point* median filter, and the sequence  $X$  is invariant to it. The term *invariant* means that the features of  $X$  have been preserved undistorted in  $A$ . If a three-point linear mean filter is applied to  $X$ , then the step discontinuity is transformed into a ramp in the resulting sequence  $B = \{\emptyset, 6, 6, 8, 10, 12, 12, \emptyset\}$ . The true utility of median filtering is demonstrated if we introduce an impulsive perturbation into  $X$  and consider the input

sequence  $\hat{X} = \{6, 6, 6, 6, 12, 102, 12, 12\}$ . Upon application of the three-point median filter we note that  $\hat{A} = \text{MF}[\hat{X}] = A$ : the output of the median filter is not perturbed by the presence of  $x_6 = 102$ . The output of the three-point linear mean filter  $\hat{B} = \{\emptyset, 6, 6, 8, 40, 42, 42, \emptyset\}$  contains extensive perturbations, however. To conclude this example, we observe that the difference sequence  $\hat{X} - \hat{A} = \{\emptyset, 0, 0, 0, 0, 90, 0, \emptyset\}$ . The significance of this result for the recommended algorithm is obvious, and it is also clear that the linear mean filter mask did not perform as well for this input. Because the median filter rejected the impulsive signal component while passing the step sequence, only the impulse remained in the difference sequence.

Because median filters are highly nonlinear, the determination of which output effects arise from which input components is extremely difficult. Due to the superposition principle, such analysis is relatively simple for linear systems. In the preceding example, the median filter seemed to preserve the monotonic features of the input, whereas the linear mean filter smeared them. The median filter also seemed to be capable of rejecting a "spiky" signal component better than the linear mean filter. In an effort to attach some quantitative meaning to the term "spiky," we consider the application of a median filter to a sequence of independent identically distributed random variables  $\{x_i\}$  with distribution  $f(x)$  and density  $F(x) = \int_{-\infty}^x f(x) dx$ . The median of these variables is that value  $v$  for which  $F(v) = 1/2$ . The ability to eliminate low probability high power impulses from the input is a well known property of median filters. Furthermore, such impulses cannot be removed by linear systems [48]. The presence of these impulses is characteristic of inputs whose probability density functions exhibit large tails. The usefulness of the median filter in smoothing applications is primarily due to the insensitivity of the median to the tails of heavily-tailed density functions [22,45]. Further discussion of the statistical properties of median filters will be deferred until Section 2.2.3.

Median filters with even sized windows have not been treated in the open literature. The analysis of such filters is considerably more difficult than that of median filters with odd sized windows for two reasons. First, a connected window containing an even number of samples cannot be symmetric. Secondly, the averaging operation inherent in computing the median of an even number of samples introduces pseudo-linear effects in the filtered output. Unless specifically stated otherwise, we consider only median filters with odd sized windows in the remainder of this chapter.

A virtually limitless variety of window sizes and shapes could theoretically be used for two-dimensional median filtering. For the remainder of this thesis, we consider only symmetric square windows. Since we are assuming that the number of samples in the window is odd, this implies that the two-dimensional windows will be of odd length in both the horizontal and vertical directions. The deterministic properties of median filters will be examined in Section 2.2.2.

## 2.2.2 Deterministic Properties

As was stated in Section 2.2.1, the set  $R$  of local order statistics is one-dimensional for any median filter, irrespective of the dimension of the filter. Consequently, one- and two-dimensional median filters can be treated together in many respects. In the following discussions, we let the symbol  $N$  denote the size of the filter window in samples and assume that  $N$  is odd. For a one-dimensional filter,  $N$  is the length of the window, while for a two-dimensional filter with an  $M \times M$  window,  $N = M^2$ . A unit impulse frame is shown in Figure 4(a). Ignoring ambiguities at the edges of the frame, the impulse response of *any* two-dimensional median filter with  $N \geq 3$  is an all zero frame.

The impulse response of *all* one-dimensional median filters with  $N \geq 3$  is also zero. Figure 4(b) shows a unit edge frame. This input is invariant to the  $M \times M$  median filter, as the output is also a unit edge frame. Likewise, the unit step sequence is invariant to all one-dimensional median filters with symmetric windows. Since the unit step and unit edge are sums of appropriately shifted unit impulses, these examples illustrate the fact that the principle of superposition does not hold for median filters. Hence, median filters are nonlinear.

Two fundamental properties of median filters have been noted by many authors. First, the scaling property states that for any monotonic function  $g(\cdot)$ ,  $\text{MF}[g(\cdot)] = g(\text{MF}[\cdot])$  [29,40,45]. Secondly, the median of a monotonic subsequence is the middle element in both value and position. Consequently, monotonic sequences are invariant to two-dimensional  $M \times M$  median filters and one-dimensional  $N$ -point median filters. Before examining other properties, we present several definitions. The term *neighborhood* is used to mean any contiguous group of samples.

We define a *constant neighborhood* as a contiguous group of samples, all of which have the same value. In one dimension, we require that the extent of the neighborhood be at least  $\lceil (N+1)/2 \rceil$  samples. In the two-dimensional case, we require that the extent of the neighborhood be everywhere at least  $\lceil (M+1)/2 \rceil$  samples in each dimension.

In one dimension, we define a sequence  $\{x_n\}$  to be *locally monotonic* with length  $k$  if and only if the neighborhood  $\{x_i, \dots, x_{i+k-1}\}$  is monotonic for all  $i$ . Such a sequence is also locally monotonic with all lengths less than  $k$ . The definition is somewhat more cumbersome in two dimensions, and is dependent on the particular filter window under consideration. Hence, two-dimensional neighborhood monotonicity is always defined with respect to a particular filter window. First, we center the window on the origin of

```

0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

```

(a)

```

1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

```

(b)

**Figure 4. Simple frames:** (a) unit impulse frame; (b) unit edge frame.

---

the real plane  $\mathbb{R} \times \mathbb{R}$ , and consider the set of all line segments which are completely contained by the window and also pass through the origin. We denote this set of line segments with the symbol  $L$ . Letting  $\mathbb{Z}$  denote the integers, we restrict  $L$  to  $\mathbb{R} \times \mathbb{R} \cap \mathbb{Z} \times \mathbb{Z}$ . The two-dimensional neighborhood contained by the window is monotonic with respect to the window if and only if, for each element  $\ell$  in the restriction of  $L$ ,  $\ell$  is monotonic. If all neighborhoods in a frame are monotonic with respect to a particular window, then we say that the frame is *locally monotonic* with respect to that window.

We formally define an *impulse* as the occurrence of fewer than  $\lfloor (N+1)/2 \rfloor$  contiguous extreme samples in the interior of a neighborhood that would otherwise be constant or monotonic. By extreme, we mean that the impulsive samples must have greater amplitudes than all other samples in the neighborhood. In one dimension, we take interior to mean that any two impulses must be separated from one another by at least  $\lfloor (N+1)/2 \rfloor$  samples. In two-dimensions, we require impulses to be separated from one another on all sides by at least  $\lfloor (M+1)/2 \rfloor$  samples.

We define an *edge* as any boundary between constant or monotonic neighborhoods. We define an *oscillation* as two or more contiguous samples that are not part of a constant neighborhood, a monotonic neighborhood, an edge, or an impulse. Finally, we refer to any sequence that is invariant to a specific median filter as a *root* of that filter.

To investigate what types of signals are roots of median filters, we observe that by our definition any sequence which is locally monotonic must be composed entirely of monotonic and constant neighborhoods. Furthermore, in a locally monotonic sequence any two monotonic neighborhoods that are opposite in trend must be separated by a

constant neighborhood. That a sequence which is everywhere locally monotonic is invariant to median filters has been proven by Tyan, who called such roots *type I* [45]. In one dimension, a sequence that is locally monotonic with length  $k$  is a type I root of a median filter of length  $N$ ,  $N \leq 2k - 3$ . In two dimensions, a sequence that is locally monotonic with respect to a given window is a type I root of any median filter that employs that window, or a connected subset of that window. Tyan also observed the existence of *type II* roots which are nowhere locally monotonic [45]. He found the theoretical treatment of type II roots difficult (especially so in two dimensions), but was able to prove that any such root must be composed of oscillatory neighborhoods. He also showed that each sample of a type II root must assume one of exactly two distinct values. As an example, consider the infinite one-dimensional sequence  $\{ \dots, -1, +1, -1, +1, \dots \}$ . This sequence is not locally monotonic with any length greater than one, yet it is a root of a median filter with  $N=5$  or  $N=9$ . It is *recurrent* to a three-point median filter, since it is invariant to any even number of repeated applications of the filter. Note, however, that it is not a root of the three-point median filter.

Another proof that locally monotonic inputs are invariant to median filtering was given by Gallagher and Wise [42]. They circumvented the perplexities associated with type II roots by considering only finite inputs, and adopting a convention of appending constant samples to the input at points where a complete neighborhood did not exist. Consequently, in their proofs they were always able to deduce the first filtered output and proceed using inductive arguments. Gallagher and Wise also formalized the concept of a passband for median filters [42]. Input features consisting of constant and monotonic neighborhoods are not attenuated by a median filter, and hence are in the



filter passband. In this respect, every neighborhood in a root sequence must either be in the filter passband or posses type II characteristics.

### 2.2.3 Statistical Properties

First we will consider median filtering the input sequence  $X$  where the  $x_i \in X$  in the one-dimensional case, or the  $x_{i,j} \in X$  in the two-dimensional case, are stationary independent identically distributed random variables with mean  $\mu$ , variance  $\sigma^2$ , distribution  $F_X(x)$  and density  $f_X(x)$ . Since the inputs are independent and the set  $R$  of order statistics is always one-dimensional, the density of  $Y = \{v_i\}$ , the running medians output by a median filter of size  $N$ , does not depend on the dimension of the input. Ataman, Aatre, and Wong [41] have stated that for  $N = 2k + 1$ , the density of  $v_i \in Y$  is:

$$f_Y(v) = \frac{N!}{k!k!} f_X(v) [F_X(v)]^k [1 - F_X(v)]^k \quad (2.2.7)$$

Equation 2.2.7 was also stated by Justusson, who used a slightly different expression for the leading coefficient [43]. A proof due to Papoulis is given on page 175 of reference [49]. Kuhlmann and Wise obtained a closed form expression for the density function  $F_Y(v)$  of the medians by integrating Equation 2.2.7 [40]. For large  $N$  and  $f_X(x)$  symmetrically distributed about  $\mu$ , Justusson [43] gave an expression for  $\sigma_v^2$ , the variance of the medians, as:

$$\sigma_v^2 = \frac{1}{4Nf_X^2(v)} \quad (2.2.8)$$

In reference [43], he also gave an approximation to Equation 2.2.8 which provides an improved estimate of  $\sigma_v^2$  when  $N$  is small. Narendra pointed out that if a linear mean

filter were employed, then the variance of the running means,  $\sigma_\mu^2$ , would not depend on the parent density  $f_x(x)$  [35]. From Equation 2.2.8, it is clear that  $\sigma_v^2$  is a function of the parent density. This fact has significance for the recommended algorithm. In particular, Justusson showed that when  $F_x(x)$  was a relatively "spiky" double exponential distribution, the variance of  $v_i$  was 50 percent smaller than the variance of the running means [43]. Hence, the median filter is more effective than a mean filter for removing target-like impulses from the input. Narendra obtained a similar result when the input variables assumed a heavily-tailed log-normal distribution [35]. A different situation arises, however, when the inputs are allowed to assume a smoother distribution more typical of the clutter in the ATWS. Justusson showed  $\sigma_v^2$  to be 57 percent larger than  $\sigma_\mu^2$  when the parent distribution is normal [43]. This important result hints that in addition to being better at attenuating targets, the median filter also passes clutter more effectively than the linear mean filter.

Kuhlmann and Wise studied the output autocorrelation and power spectrum of median filtered sequences of independent identically distributed stationary random variables [40]. They found that pairs of output samples exhibited a nonzero covariance which was strongly dependent on the number of common samples in the windows used to compute them. Their results corroborate the assertion that median filters tend to greatly attenuate small, high spatial frequency features in the input. Justusson presented a similar result, and also studied the responses of median and mean filters to nonindependent normally distributed inputs [43]. He found that when the inputs exhibited nonnegative correlations, the output variances of median and mean filters were related by:

$$1 \leq \frac{\sigma_v^2}{\sigma_\mu^2} \leq \frac{\pi}{2} \quad (2.2.9)$$

This result once again hints at the ability of median filters to preserve non-impulsive input features better than linear averaging filters.

Next we consider median filtering sequences of independent stationary random variables that are not identically distributed. Ataman, Aatre, and Wong investigated the output of three- and five-point median filters when some input samples had Gaussian distributions, while others had heavily-tailed "spiky" distributions [41]. They found that the presence of Gaussian inputs somewhat impaired the ability of the median filter to completely remove impulses from the input, but that the median filter was still superior to a linear Hanning filter for this purpose.

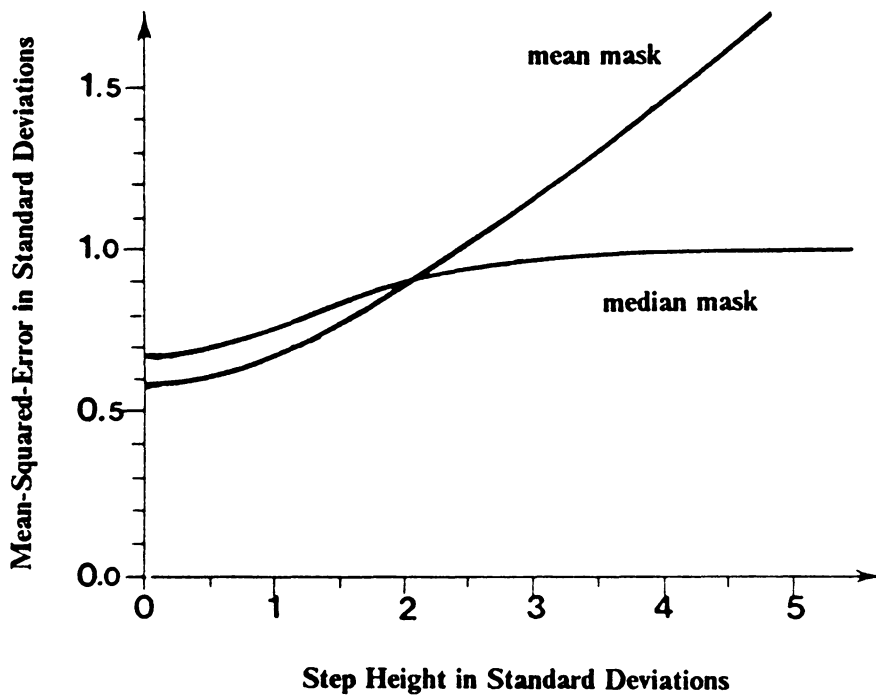
By examining the output variance, we have suggested in the previous discussions that the median filter is better able to pass clutter features undistorted than comparable linear smoothers. That this is indeed the case will now be demonstrated. In Section 1.2 we stated that the clutter in the ATWS is assumed to be strictly greater in extent than the targets. Although the clutter may contain many high spatial frequencies in the form of sharp edges, it can not contain impulses in the sense that they were defined in Section 2.2.2. The occurrence of such an impulse in an input frame is always assumed to arise from a target. Consider as input the one-dimensional step sequence  $\{x_i\}$  or two-dimensional edge frame  $\{x_{i,j}\}$  with height  $h$  and size  $n$  samples that has been corrupted by uncorrelated noise. The input samples on the low side of the step or edge are independent stationary random variables normally distributed with zero mean and variance  $\sigma^2$ . On the high side, they are normally distributed with mean  $h$  and variance  $\sigma^2$ . In one dimension, we denote the uncorrupted step sequence by  $\{s_i\}$  and define the mean-squared-error in the filtered output sequence  $\{y_i\}$  by:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n E[(y_i - s_i)^2] \quad (2.2.10)$$

where  $E[.]$  is the expectation operator. The mean-squared-error is similarly defined in two dimensions. Justusson has numerically solved for the mean-squared-error in the output of both one- and two-dimensional median and mean filters [43]. He found that for  $h > 3\sigma$  the MSE of the median filter output was significantly lower than that of the linear filter. A graph of his result for the MSE of one-dimensional three-point filters as a function of  $h$  is shown in Figure 5, where both axes are in units of standard deviations. In a similar experiment where the step was further corrupted by impulsive noise, Ataman, Aatre, and Wong found that not only was the median filter MSE lower than that of a linear Hanning filter, but that the median filter was also far better able to remove the impulses despite the presence of the normally distributed noise [41]. We conclude this section by noting that in a one-dimensional analysis Nades and Gallagher derived an expression for the output distribution of a median filter in response to any general stochastic signal [48]. They found that the MSE of the median filter was orders of magnitude smaller than that of a linear averaging filter when the input contained additive impulses.

## 2.2.4 Frequency Domain Characterization

Because median filters are highly nonlinear, their frequency domain analysis is difficult [29]. Although frequency domain analysis is extremely important in the study of linear



**Figure 5. Mean-squared-error in median and mean filtered noisy step (from reference [43])**

systems, it is of less utility in the analysis of median filters because the superposition principle does not hold for the latter. Nevertheless, a few researchers have studied the response of median filters to sinusoidal inputs. Justusson derived the power spectrum distribution of median filtered cosine waves in both one and two dimensions [43]. Velleman studied the power transfer characteristics of median filters in response to sinusoidal inputs, and found that they often contain significant sidelobes [44]. He suggested that the sidelobes are strongly related to the roots and recurrent sequences of the filter. In reference [43], Justusson stated some results of Heygster, who constructed empirical transfer functions for median filters by taking quotients of the Fourier transforms of specific two-dimensional input and output sequences. The only general result obtained from these analyses was that the spectral responses of running medians are similar to those of running means for frequencies  $\omega_0 \leq 2\pi/5$  [43].

### 2.2.5 Threshold Decomposition

The purpose of this section is to conclude our discussion of median filter properties by citing the existence of a theory for median filters that is in many respects analogous to the superposition principle of linear systems. Although the theory is not applicable to the ATWS prefilter problem, it is presented for completeness. The theory was developed by Fitch, Coyle, and Gallagher, and is called *threshold decomposition* [38]. It has only been developed for one-dimensional filters. Several researchers have concerned themselves with the convergence of arbitrary inputs to roots upon repeated applications of a nonrecursive median filter, or upon the single application of a recursive median filter [37,38,39,42,45,46].

A recursive median filter is one for which each output sample is used in the computation of subsequent output samples. For example, consider median filtering the sequence  $\{x_1, x_2, x_3, x_4\}$  with a three-point filter. By our endpoint convention,  $y_1$  and  $y_4$  are indeterminate. For both a recursive and a nonrecursive median filter,  $y_2 = \text{MF}[\{x_1, x_2, x_3\}]$ . With a nonrecursive filter,  $y_3 = \text{MF}[\{x_2, x_3, x_4\}]$ . However, for the recursive filter  $y_3 = \text{MF}[\{y_2, x_3, x_4\}]$ .

A well known result is that successive applications of a nonrecursive median filter will reduce any arbitrary input sequence to a root. Gallagher and Wise proved that any finite one-dimensional input sequence of length  $L$  samples will be reduced to a root of any one-dimensional median filter after at most  $\frac{1}{2}(L-2)$  applications of the filter [42]. Another well known result is that any one-dimensional input sequence will be reduced to a root after one application of a recursive median filter. Although this result is often true in two dimensions as well, Nodes and Gallagher have constructed two-dimensional inputs for which it does not hold [46].

Frequency domain analysis of linear systems is based on the fact that any input can be decomposed into a linear combination of some set of basis signals. An orthonormal set of complex exponentials is often chosen as the basis. The system response is then determined as a linear combination of the responses to the individual basis signals. For median filters, any input with  $k$  quantization levels can be decomposed into a sum of binary sequences. Suppose that repeated application of a specific median filter would reduce the input sequence  $X$  to the root sequence  $Y_r$ . The threshold decomposition theory states that if the binary sequences composing  $X$  were independently reduced to roots by repeated applications of the specified filter, then those roots could be recombined via a mapping from the space of binary sequences to the space of  $k$ -valued

sequences in such a way as to yield  $Y$ , [38]. This theory is useful for describing the convergence of input sequences to root sequences, and Arce has extended it to include recursive median filters [39].

## 2.3 Formalization of the Recommended Algorithm

In this section, three specific variants of the recommended algorithm will be presented. They will be evaluated in Chapter Three. Before presenting them, we note that the general form of the recommended algorithm has appeared previously in the open literature. Justusson described the technique of *residual smoothing* for recovering the signal  $\{s_i\}$  from the noise-corrupted input  $\{x_i\} = \{s_i\} + \{n_i\}$  [43]. In using residual smoothing, one constructs a noise estimate  $\{\hat{n}_i\} = \{x_i\} - \text{MF}[\{x_i\}]$  and a correction factor  $\{z_i\} = \text{MF}[\{\hat{n}_i\}]$ . The signal is then estimated by  $\{\hat{s}_i\} = \text{MF}[\{x_i\}] + \{z_i\}$ . Of interest is the fact that the noise estimate  $\{\hat{n}_i\}$  is computed with the recommended algorithm. Justusson also made a brief specific reference to the use of the recommended algorithm for object extraction [43]. He gave an example comparing a  $3 \times 3$  mean filter mask to a  $3 \times 3$  median filter mask.

Tyan investigated smoothing algorithms of the form

$$Y = \sum_{k=0}^n a_k H^k[X] \quad (2.3.1)$$



where  $H^k[X]$  denotes  $k$  repeated applications of the smoothing operator  $H$  to the input  $X$  [45]. In particular, Tyan used median filters as smoothing operators and required that  $a_0 \neq 1$ . However, the recommended algorithm can be derived from Equation 2.3.1 by letting  $n = a_0 = 1$  and  $a_1 = -1$ .

### 2.3.1 Variant I: $3 \times 3$ Median Filter Mask

Based on the theoretical arguments presented in Section 2.2, we expect that some two-dimensional median filter should be well suited for use as the masking operator in the recommended algorithm. The geometries of the three target models used in this thesis were shown in Figure 1. We must choose a median filter for which these target smears fall outside the passband. To pass clutter features with minimum distortion, we would also like the median filter to have as many roots as possible, provided that the target smears are not roots.

Since input features less than  $[(N + 1)/2]$  pixels in extent are eradicated by a median filter with  $N$  pixels in its window, the number of roots decreases monotonically with the window size (*all* possible input sequences are roots of a one-point median filter). Since the largest target smear is four pixels in extent, we require that  $N > 8$ . The smallest square symmetric window satisfying this constraint is of size  $3 \times 3$ , giving  $N = 9$ . Stated another way, this is the smallest two-dimensional median filter to which the targets will appear impulsive. We refer to the recommended algorithm with a  $3 \times 3$  median filter masking operator as  $RA_1$ , which stands for variant one of the recommended algorithm.

### 2.3.2 Variant II: Compensated Median Filter Mask

We now propose another variant of the median filter unsharp masking algorithm by making a modification to  $RA_I$ . Reasoning qualitatively, we observe that the amplitude of a neighborhood median might be biased upwards by either the simultaneous presence of several target pixels in the window, or by the simultaneous presence of heavily-tailed noise and one or more target pixels. This line of reasoning is somewhat corroborated by the fact that, as was alluded to in Section 2.2.3, the expected value of an input impulse at the output of a median filter is nonzero when uncorrelated noise is present. Such upward biasing of the neighborhood median away from the true value of the local clutter median is a detrimental effect in the ATWS prefilter, since we wish to subtract as small a value as possible from target pixels. To compensate for this upward bias, we will not include the center pixel of the window when computing the neighborhood median. This compensation method was suggested to the author by researchers in the Optical Sciences Division of the United States Naval Research Laboratory. Incorporating a  $3 \times 3$  compensated median filter masking operator into the recommended algorithm, the filtered outputs are computed as:

$$y(i, j) = x(i, j) - MF[\{x(i-1, j-1), x(i-1, j), x(i-1, j+1), x(i, j-1), \\ x(i, j+1), x(i+1, j-1), x(i+1, j), x(i+1, j+1)\}] \quad (2.3.2)$$

We refer to this variant of the recommended algorithm as  $RA_{II}$ .

### 2.3.3 Variant III: One-Dimensional Median Filter Mask

In a widely cited study, Narendra investigated the use of separable median filters for noise removal in infrared imagery [35]. Since one-dimensional filters are much easier to implement than two-dimensional filters, he applied a one-dimensional median filter first to the columns of an image, and then to the rows. He found that the noise cleaning capability of the separable filter was quite comparable to that of the two-dimensional filter, and that the output variance of the separable filter was always a little greater than that of its two-dimensional counterpart. With reference to the recommended algorithm, a slightly greater output variance is desirable since it is indicative of improved clutter tracking ability. Furthermore, we do not expect clutter to leak through a one-dimensional median filter mask near the junctions of edges. For example, the corners of a square of any size do not appear locally monotonic to a two-dimensional median filter. To a one-dimensional filter, however, they appear perfectly monotonic provided that the square extends for at least  $\lceil (N + 1)/2 \rceil$  pixels along each edge and is not rotated with respect to the edges of the frame.

As a third variant of the recommended algorithm, we consider a masking operator which involves the application of a one-dimensional median filter to only the rows of the input frame. We reason that this mask should perfectly preserve vertical clutter features, and that horizontal features should not be distorted any more than they would be by a comparable two-dimensional median filter. Although these arguments are true, the disadvantage of using a one-dimensional filter mask is that it must be of length  $N \geq 5$  to prevent the two- and four-pixel target smears from being roots. Hence, horizontal clutter features will be distorted more by the one-dimensional filter mask than by the

two-dimensional masks used for  $RA_I$  and  $RA_{II}$ . In mathematical form, the output of this variant to the recommended algorithm is:

$$y(i, j) = x(i, j) - \text{MF}[\{x(i, j - 2), x(i, j - 1), x(i, j), x(i, j + 1), x(i, j + 2)\}] \quad (2.3.3)$$

We refer to this prefilter as  $RA_{III}$ .

## 3 Algorithm Evaluation

In this chapter, the spatial prefiltering algorithms presented in Chapters One and Two are quantitatively evaluated. Section 3.1 provides an explanation of the criteria against which the algorithms are measured. A simple computer experiment is described in Section 3.2. The frames used for this experiment are small enough to be presented as figures in this thesis. Finally, in Section 3.3 the variants of the recommended algorithm are pitted against the linear point detection filter in an experiment involving realistic input imagery for the ATWS.

### *3.1 Measurement Criteria*

Generally accepted methods of evaluating automatic target detection systems do not exist [4]. Consequently, attempts to compare algorithms based on results cited in the open literature are difficult and often fruitless. One metric that appears to be fairly common is the enhancement of the signal to clutter ratio. *Signal to clutter ratio* is

usually defined as the target energy (amplitude) present in a frame divided by the standard deviation of the clutter. The enhancement of this ratio due to a specific filtering operation is computed by dividing the signal to clutter ratio after filtering by the signal clutter ratio before filtering. In this thesis, the enhancement of the signal to clutter ratio will be represented with the symbol SCE defined according to:

$$\text{SCE} = \frac{S_f \sigma_u}{S_u \sigma_f} \quad (3.1.1)$$

where

- $S_f$  = target energy in filtered frame
- $\sigma_f$  = filtered clutter standard deviation
- $S_u$  = target energy in unfiltered frame
- $\sigma_u$  = unfiltered clutter standard deviation

$S_u$  is computed by summing the amplitudes of all targets in the input frame, while  $S_f$  is computed by summing target amplitudes in the output frame. As was mentioned in Section 2.2.1, it is often impossible to separate precisely median filtered output into those components that arise from targets in the input and those components that arise from the clutter. Since the simulated targets in this thesis are created by adding excess amplitude to certain pixels of a frame, only the added amplitude is included in the summation when  $S_u$  is calculated. To a rough approximation, we expect the amplitude of output components arising from clutter to be nearly zero. Consequently, we will include the total amplitude present at all target pixels in the summation when calculating  $S_f$ . The ratio  $S_f/S_u$  is a relative measure of how much of the target energy at the prefilter input is preserved in the prefilter output. This ratio increases monotonically as more target energy is passed by the prefilter, and it equals one if and only if all target energy is passed unattenuated.

Irrespective of whether an input frame or an output frame is under consideration, the clutter standard deviation is always calculated as the square root of the clutter variance, which is computed from only those pixels that do not contain a target and are not in positions that will be indeterminate in the filter output. Formally, the clutter mean of the frame  $X$  is defined by:

$$\bar{x}_c = \frac{1}{k} \sum_i \sum_j x_{i,j} \quad (3.1.2)$$

where the sums are taken over all pixels that are not targets and do not assume indeterminate values in the filter output. The integer  $k$  is the number of such pixels in the frame. The clutter variance of the frame  $X$  is then defined as

$$\sigma_c^2 = \frac{1}{k-1} \sum_i \sum_j (\bar{x}_c - x_{i,j})^2 \quad (3.1.3)$$

where once again the sums are taken over only those pixels that are not targets and do not assume indeterminate values in the filter output. It is a well known fact from statistics that when an entire population is not available, an improved estimate of the variance is obtained by taking the denominator of the leading coefficient equal to one less than the number of samples in the sample population. Since we do not include those clutter pixels at the edges of frames or at targets in the summations of Equation 3.1.3, the denominator of the leading coefficient is taken equal to  $k - 1$  to provide as accurate an estimate as possible of the true clutter variance. Note that the ideal algorithm has  $SCE = \infty$ , since the clutter variance is reduced to zero at the filter output. As an algorithm evaluation metric, SCE favors clutter rejection capability more than it favors

target passing ability. This is true because the ratio  $S_f/S_u$  has a maximum value of one, which is realized with the ideal algorithm. The ratio  $\sigma_u/\sigma_f$  can become quite large, however. This ratio is unbounded when the ideal algorithm is evaluated. Consequently, we seek a second evaluation metric that is a stronger function of target passing ability.

For use in this thesis, an algorithm measurement criterion that we shall call the *beta factor* is proposed. It is defined by:

$$\beta = \frac{S_f/S_u}{1 + \frac{\text{MSE}_c}{\sigma_u^2}} \quad (3.1.4)$$

where

- $S_f$  = target energy in filtered frame
- $S_u$  = target energy in unfiltered frame
- $\text{MSE}_c$  = clutter mean-squared-error, see below
- $\sigma_u^2$  = unfiltered clutter variance

We want a prefilter that, like the ideal algorithm, rejects all clutter. Hence the desired output at all non-target pixels is zero. We consider any nonzero output at such pixels to be an error in the sense that the prefilter was not able to perfectly track the clutter features. In Equation 3.1.4, the clutter mean-squared-error,  $\text{MSE}_c$ , is computed by summing the squares of all nonzero amplitudes among the non-target pixels in the output frame and dividing this sum by the number of non-target pixels in the input frame. It is a positive definite monotonically increasing function of the total amount of clutter amplitude that leaks through the filter.  $\text{MSE}_c$  is zero if and only if all clutter features are perfectly suppressed. To allow comparison between frames with widely different clutter variances,  $\text{MSE}_c$  is scaled by the unfiltered clutter variance in Equation



3.1.4. We observe that the ideal algorithm has  $\beta = 1$ . In some sense,  $\beta$  is a percentage measure of how well a given filter performs with respect to the ideal algorithm.

## 3.2 *A Simple Experiment*

In this section, six prefiltering algorithms are evaluated on simple input frames. The six algorithms are the Laplacian filter, the point detection filter, unsharp masking with a  $3 \times 3$  linear averaging mean filter mask, and the three variants of the recommended algorithm presented in Section 2.3. For this experiment, we are more interested in evaluating the theoretical performance of the algorithms than in evaluating any particular hardware implementation. Hence, quantization and finite word length effects are not explicitly considered. We use floating-point numbers to represent the pixels of each frame, and allow the filters to perform floating-point arithmetic. Finite word length effects will be considered in Section 3.3. Quantization effects are of no particular interest in this thesis: we define targets statistically by their amplitudes in the frames coming out of the sensor subsystem. Targets are not defined prior to quantization in the image formation process. To measure the clutter rejection capabilities of these algorithms, we use the input frames shown in Figures 6 and 7. The input frame of Figure 6 is a unit edge frame with unfiltered clutter standard deviation  $\sigma_u = 5.000 \times 10^{-1}$ . The input frame of Figure 7 is all zero, except for a  $5 \times 5$  square of unit amplitude pixels at the center. This frame has unfiltered clutter standard deviation  $\sigma_u = 4.648 \times 10^{-1}$ . To examine the ability of the six algorithms to pass targets while attenuating clutter, we begin with the frame shown in Figure 8(a). This frame consists only of smooth, monotonic background. The same input is shown again in Figure 8(b) with four bright targets

added. The top two targets are blurred across four pixels, while the bottom two are true point targets. The input frame of Figure 8(b) has  $\sigma_u = 1.765$ ,  $S_u = 90$ , and  $S_u/\sigma_u = 51.001$ . The results of prefiltering the input frames of Figures 6, 7, and 8(b) are discussed in the remainder of this section, where -99.0 is used to represent an indeterminate filter output. For convenience, the  $\beta$  and SCE data obtained from Figure 8(b) are also shown in Table 1.

### 3.2.1 $3 \times 3$ Laplacian Filter

The  $3 \times 3$  Laplacian filter mask was shown in Equation 1.3.3. To realize a prefilter for the ATWS using this mask, the mask is convolved with the input frame using Equation 1.3.1. The Laplacian filter response to the unit edge frame of Figure 6 is shown in Figure 9(a). As the output is an all zero frame,  $MSE_c = 0$  and  $\sigma_u/\sigma_f = \infty$ . Next we apply the Laplacian filter to the square input frame of Figure 7. The response is shown in Figure 9(b), and we see that 16 pixels are nonzero. This gives  $MSE_c = 1.778 \times 10^{-2}$ , and  $\sigma_u/\sigma_f = 3.465$ .

The response of the Laplacian filter to the input frame of Figure 8(b) is shown in Figure 9(c). We see that although the point targets have been passed perfectly, the blurred targets have been severely attenuated. This situation arises because the -2 coefficients in the Laplacian filter mask result in target energy being subtracted from target energy in the filtered output. In addition, we note significant clutter leakage around the targets. The beta factor of the Laplacian filter for this input is  $2.255 \times 10^{-1}$ .  $SCE = 4.590 \times 10^{-1}$ , which indicates that the signal to clutter ratio in the Laplacian filter output is actually lower than the signal to clutter ratio in the input frame. Although it is quite adept at

```
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

**Figure 6. Unit edge frame**

---

```
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

**Figure 7. Input frame with  $5 \times 5$  unit amplitude square**

---

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0

(a)

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	10.0	10.0	1.0	1.0	1.0	2.0	12.0	13.0	5.0	6.0	7.0
1.0	1.0	1.0	10.0	10.0	1.0	1.0	1.0	2.0	12.0	13.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	10.0	1.0	1.0	1.0	1.0	2.0	12.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0

(b)

Figure 8. Smooth monotonic background frame: (a) without targets; (b) with four bright targets added.

**Table 1. SCE and Beta for the input of Figure 8(b)**

	<u>Beta</u>	<u>SCE</u>
Laplacian Filter.....	0.225	0.459
Point Detection Filter.....	0.546	1.577
Mean Filter Unsharp Masking...	0.513	1.601
RA I.....	0.946	9.872
RA II.....	0.952	8.232
RA III.....	0.867	4.763

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(a)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.3	-0.3	0.0	0.0	0.0	-0.3	0.3	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.3	0.0	0.0	0.0	0.3	-0.3	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.3	0.0	0.0	0.0	0.3	-0.3	0.0	0.0	-99.0
-99.0	0.0	0.3	-0.3	0.0	0.0	0.0	-0.3	0.3	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(b)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	2.3	-2.3	-2.3	2.3	0.0	0.0	2.3	-2.3	-2.3	2.3	0.0
-99.0	0.0	-2.3	2.3	2.3	-2.3	0.0	0.0	-2.3	2.3	2.3	-2.3	0.0
-99.0	0.0	-2.3	2.3	2.3	-2.3	0.0	0.0	-2.3	2.3	2.3	-2.3	0.0
-99.0	0.0	2.3	-2.3	-2.3	2.3	0.0	0.0	2.3	-2.3	-2.3	2.3	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	2.3	-4.5	2.3	0.0	0.0	0.0	2.3	-4.5	2.3	0.0	0.0
-99.0	0.0	-4.5	9.0	-4.5	0.0	0.0	0.0	-4.5	9.0	-4.5	0.0	0.0
-99.0	0.0	2.3	-4.5	2.3	0.0	0.0	0.0	2.3	-4.5	2.3	0.0	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(c)

Figure 9. Laplacian filter responses: (a) unit edge response; (b) response to square input; (c) response to the input of figure 8(b).

rejecting clutter, we see that the Laplacian filter is not able to pass multi-pixel target smears effectively. Consequently, it will not be considered in the experiments of Section 3.3.

### 3.2.2 $3 \times 3$ Point Detection Filter

The point detection filter mask was shown in Equation 1.3.4, and it is applied using Equation 1.3.1. Figure 10(a) shows the point detection filter response to the unit edge frame of Figure 6. The output is nonzero for 18 pixels, yielding  $MSE_c = 3.556 \times 10^{-2}$ , and  $\sigma_u/\sigma_f = 2.635$ . When the point detection filter is applied to the square input of Figure 7, the result shown in Figure 10(b) is obtained. The number of nonzero output pixels is 40,  $MSE_c = 7.457 \times 10^{-2}$ , and  $\sigma_u/\sigma_f = 1.692$ . Hence, the point detection filter does not reject clutter as well as the Laplacian filter. However, it passes targets much better than the Laplacian filter.

The point detection filter response to the input frame of Figure 8(b) is shown in Figure 10(c). Due to the symmetric configuration of -1 coefficients in the filter mask, the blurred targets are attenuated less than they were by the Laplacian filter. We also note that the point detection filter output contains less clutter leakage around the targets than did the Laplacian filter output. The beta factor of the point detection filter for this input is  $5.464 \times 10^{-1}$ , and  $SCE = 1.577$ . Although these results are not particularly noteworthy, the point detection filter will be further investigated for comparative purposes in Section 3.3.



-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	-99.0
-99.0	-0.4	-0.4	-0.4	-0.4	-0.4	-0.4	-0.4	-0.4	-0.4	-0.4	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(a)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-0.1	-0.3	-0.4	-0.4	-0.4	-0.3	-0.1	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.6	0.4	0.4	0.4	0.6	-0.3	0.0	0.0	-99.0
-99.0	0.0	-0.4	0.4	0.0	0.0	0.0	0.4	-0.4	0.0	0.0	-99.0
-99.0	0.0	-0.4	0.4	0.0	0.0	0.0	0.4	-0.4	0.0	0.0	-99.0
-99.0	0.0	-0.4	0.4	0.0	0.0	0.0	0.4	-0.4	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.6	0.4	0.4	0.4	0.6	-0.3	0.0	0.0	-99.0
-99.0	0.0	-0.1	-0.3	-0.4	-0.4	-0.4	-0.3	-0.1	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(b)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.4	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-1.1	-2.3	-2.3	-1.1	0.0	-0.4	-1.1	-2.3	-2.3	-1.1	0.0	-99.0
-99.0	0.0	-2.3	5.6	5.6	-2.3	0.0	-0.4	-2.3	5.6	5.6	-2.3	0.0	-99.0
-99.0	0.0	-2.3	5.6	5.6	-2.3	0.0	-0.4	-2.3	5.6	5.6	-2.3	0.0	-99.0
-99.0	0.0	-1.1	-2.3	-2.3	-1.1	0.0	-0.4	-1.1	-2.3	-2.3	-1.1	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.4	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.4	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-1.1	-1.1	-1.1	0.0	0.0	-0.4	-1.1	-1.1	-1.1	0.0	0.0	-99.0
-99.0	0.0	-1.1	9.0	-1.1	0.0	0.0	-0.4	-1.1	9.0	-1.1	0.0	0.0	-99.0
-99.0	0.0	-1.1	-1.1	-1.1	0.0	0.0	-0.4	-1.1	-1.1	-1.1	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.4	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.4	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(c)

Figure 10. Point detection filter responses: (a) unit edge response; (b) response to square input; (c) response to the input of figure 8(b).

### 3.2.3 Unsharp Masking with Mean Filter Mask

This algorithm is a realization of Equation 2.2.1 with  $G(\cdot) \equiv 1$  and  $\alpha \equiv 0$ . A  $3 \times 3$  mean filter is used for the masking operator. Each output pixel  $y_{i,j}$  is computed by subtracting from  $x_{i,j}$  the mean of the  $3 \times 3$  neighborhood centered about  $x_{i,j}$ . The response of this prefilter to the unit edge frame of Figure 6 is shown in Figure 11(a). The output contains 18 nonzero pixels, giving  $MSE_e = 2.000 \times 10^{-2}$  and  $\sigma_u/\sigma_f = 3.514$ . The response to the square input of Figure 7 is shown in Figure 11(b). In this case, the output contains 37 nonzero pixels,  $MSE_e = 4.889 \times 10^{-2}$ , and  $\sigma_u/\sigma_f = 2.090$ .

The mean filter unsharp masking response to the input of Figure 8(b) is shown in Figure 11(c). For this input,  $SCE = 1.601$  and  $\beta = 5.128 \times 10^{-1}$ . Although these results are comparable to those obtained with the point detection filter, mean filter unsharp masking will not be considered in Section 3.3 for reasons of economy. As with the Laplacian filter, the primary drawback of this algorithm is that target energy is subtracted from target energy when the input contains a multi-pixel target smear.

### 3.2.4 Recommended Algorithm, Variant I

The response of  $RA_1$  to the unit edge frame of Figure 6 is shown in Figure 12(a). The clutter is perfectly rejected, yielding  $MSE_e = 0$  and  $\sigma_u/\sigma_f = \infty$ . The response of  $RA_1$  to the square input of Figure 7 is shown in Figure 12(b). There are four nonzero output pixels,  $MSE_e = 4.938 \times 10^{-2}$ , and  $\sigma_u/\sigma_f = 2.132$ . The nonzero output pixels arise from the fact that the corners of the square are not monotonic in all directions, and hence this input frame is not locally monotonic with respect to the  $3 \times 3$  median filter mask

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	-99.0
-99.0	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(a)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-0.1	-0.2	-0.3	-0.3	-0.3	-0.2	-0.1	0.0	0.0	-99.0
-99.0	0.0	-0.2	0.6	0.3	0.3	0.3	0.6	-0.2	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.3	0.0	0.0	0.0	0.3	-0.3	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.3	0.0	0.0	0.0	0.3	-0.3	0.0	0.0	-99.0
-99.0	0.0	-0.3	0.3	0.0	0.0	0.0	0.3	-0.3	0.0	0.0	-99.0
-99.0	0.0	-0.2	0.6	0.3	0.3	0.3	0.6	-0.2	0.0	0.0	-99.0
-99.0	0.0	-0.1	-0.2	-0.3	-0.3	-0.3	-0.2	-0.1	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(b)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-1.0	-2.0	-2.0	-1.0	0.0	-0.3	-1.0	-2.0	-2.0	-1.0	0.0
-99.0	0.0	-2.0	5.0	5.0	-2.0	0.0	-0.3	-2.0	5.0	5.0	-2.0	0.0
-99.0	0.0	-2.0	5.0	5.0	-2.0	0.0	-0.3	-2.0	5.0	5.0	-2.0	0.0
-99.0	0.0	-1.0	-2.0	-2.0	-1.0	0.0	-0.3	-1.0	-2.0	-2.0	-1.0	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	-1.0	-1.0	-1.0	0.0	0.0	-0.3	-1.0	-1.0	-1.0	0.0	-99.0
-99.0	0.0	-1.0	8.0	-1.0	0.0	0.0	-0.3	-1.0	8.0	-1.0	0.0	-99.0
-99.0	0.0	-1.0	-1.0	-1.0	0.0	0.0	-0.3	-1.0	-1.0	-1.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.3	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(c)

Figure 11. Responses using a mean filter mask: (a) unit edge response; (b) response to square input; (c) response to the input of figure 8(b).

employed by  $RA_I$ , (two-dimensional local monotonicity was defined in Section 2.2.2). When  $RA_I$  is applied to the input frame of Figure 8(b), the output frame shown in Figure 12(c) is obtained. Three of the four targets are perfectly extracted, since they have unattenuated amplitudes and are immediately surrounded by zero clutter leakage in the filtered output. For this input,  $SCE = 9.872$  and  $\beta = 9.465 \times 10^{-1}$ . Based on this experiment,  $RA_I$  appears to be an extremely good approximation to the ideal algorithm.

### 3.2.5 Recommended Algorithm, Variant II

In response to the unit edge frame of Figure 6, the output of  $RA_{II}$  is shown in Figure 13(a). We see that  $MSE_e = 0$  and  $\sigma_u/\sigma_f = \infty$ . The response of  $RA_{II}$  to the square input of Figure 7 is shown in Figure 13(b). As with the uncompensated median filter mask of  $RA_I$ , the corners of the square leak through to give  $MSE_e = 4.938 \times 10^{-2}$  and  $\sigma_u/\sigma_f = 2.132$ . Figure 13(c) shows the output of  $RA_{II}$  in response to the input frame of Figure 8(b). We see that  $\beta = 9.524 \times 10^{-1}$  and  $SCE = 8.232$ . For this input, the compensation has provided a slight improvement in the beta factor due to one extra amplitude unit of target energy being passed for the blurred target at the upper right corner of the input frame. Some additional clutter leakage has also been introduced around this target due to the averaging operation inherent in computing the median of an even number of samples (for  $RA_{II}$ ,  $N = 8$ ). Consequently,  $RA_{II}$  has a slightly smaller SCE than  $RA_I$  for this input. Based on this experiment,  $RA_{II}$  appears also to be an extremely good approximation to the ideal algorithm.

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(a)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(b)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0	0.0	0.0	-99.0
-99.0	0.0	0.0	9.0	9.0	0.0	0.0	0.0	0.0	8.0	8.0	-1.0	0.0	-99.0
-99.0	0.0	0.0	9.0	9.0	0.0	0.0	0.0	0.0	8.0	8.0	-1.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-1.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(c)

Figure 12. Responses of recommended algorithm variant I: (a) unit edge response; (b) response to square input; (c) response to the input of figure 8(b).

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(a)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(b)

-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	-1.0	-0.5	0.0
-99.0	0.0	0.0	9.0	9.0	0.0	0.0	0.0	0.0	8.5	8.0	-1.0	0.0
-99.0	0.0	0.0	9.0	9.0	0.0	0.0	0.0	0.0	8.5	8.0	-1.0	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	-1.0	-0.5	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	-0.5	0.0	0.0
-99.0	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	9.0	-0.5	0.0	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	-0.5	0.0	0.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0
-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0	-99.0

(c)

Figure 13. Responses of recommended algorithm variant II: (a) unit edge response; (b) response to square input; (c) response to the input of figure 8(b).

### 3.2.6 Recommended Algorithm, Variant III

Since  $RA_{III}$  utilizes a one-dimensional five-point median filter masking operator, the positions of the indeterminate pixels in the output frames of this filter are different from those in the output frames produced by the two-dimensional masking operators used for  $RA_I$  and  $RA_{II}$ . As a result, a slightly different set of pixels is used to estimate the clutter variance when  $RA_{III}$  is evaluated. In particular, with respect to  $RA_{III}$  the input of Figure 6 has  $\sigma_u = 5.012 \times 10^{-1}$ . The input of Figure 7 has  $\sigma_u = 4.713 \times 10^{-1}$ , while the input of Figure 8(b) has  $\sigma_u = 1.433$ ,  $S_u = 90$ , and  $S_u / \sigma_u = 62.822$ .

The output of  $RA_{III}$  in response to the unit edge frame of Figure 6 is shown in Figure 14(a). We observe that since  $N=5$ , there are two columns of indeterminate values on each side of the frame, and that since the filter is one-dimensional there are no indeterminate values along the top and bottom edges. For this input  $MSE_c = 0$  and  $\sigma_u / \sigma_f = \infty$ . The response of  $RA_{III}$  to the square input of Figure 7 is shown in Figure 14(b). As expected, the one-dimensional median filter mask is able to perfectly track the corners of the square, giving  $MSE_c = 0$  and  $\sigma_u / \sigma_f = \infty$ . Figure 14(c) shows the output of  $RA_{III}$  in response to the input frame of Figure 8(b).

We observe that the two targets appearing over the constant background in the left half of the input frame are passed perfectly, while the targets in the right half of the input frame are somewhat attenuated. There is also some significant clutter leakage around the latter. These effects are due to the fact that, for the monotonically increasing background in the right half of the input frame, the presence of a target in a window with a horizontal extent of five pixels biases the median upward more than in a window with a horizontal extent of three pixels. Stated another way, the five-point median filter

-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0

(a)

-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0

(b)

-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0
-99.0	-99.0	0.0	9.0	9.0	0.0	0.0	0.0	0.0	7.0	7.0	-2.0	-99.0	-99.0	
-99.0	-99.0	0.0	9.0	9.0	0.0	0.0	0.0	0.0	7.0	7.0	-2.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0	8.0	-1.0	-1.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	
-99.0	-99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-99.0	-99.0	

(c)

Figure 14. Responses of recommended algorithm variant III: (a) unit edge response; (b) response to square input; (c) response to the input of figure 8(b).



mask has "oversmoothed" the input in the horizontal direction, with the result that the targets in the smoothed image are smeared over several more pixels than they were in the original image. Upon subtraction of the smoothed image from the original, this causes significant clutter leakage around the targets. For this input,  $RA_{III}$  has  $\beta = 8.675 \times 10^{-1}$  and  $SCE = 4.763$ . As a final point, we note that the results of this experiment are not sufficient to permit any conjecture as to the effectiveness of  $RA_{III}$ . This algorithm employs an unsymmetrical window, and the clutter features in the input of Figure 8(b) are all of horizontal orientation. Had we rotated the input frame by  $90^\circ$ ,  $RA_{III}$  would have produced the same output as the ideal algorithm.

### ***3.3 Evaluation on Realistic Inputs***

In this section, an experiment comparing the linear point detection filter,  $RA_I$ ,  $RA_{II}$ , and  $RA_{III}$  is described. A total of 120 input frames were prefiltered. The United States Naval Research Laboratory provided the author with a tape containing computer generated clutter frames representative of actual imagery in the ATWS. The simulated imagery contained no targets. The author extracted 10  $128 \times 64$  frames from the tape, added one-pixel targets and two- and four-pixel target smears to these frames, and prefiltered them with each of the four algorithms. We refer to the 10 frames obtained from the tape simply as the *tape frames*. The experimental procedure is described in Section 3.3.1.

### 3.3.1 Experimental Procedure

Three experimental input frames were generated from each tape frame. An array of 72 one-pixel targets was created in the first of these. An array of 72 two-pixel target smears was created in the second, while an array of 72 four-pixel target smears was created in the third. Hence, 10 frames were created for each type of target, and a total of 30 unique experimental input frames were generated. The targets were arranged such that at most one target fell within any prefilter window at any given time. One-pixel targets were constructed by adding excess amplitude to each pixel located at a simulated target. Likewise, two- and four-pixel target smears were constructed by adding excess amplitude to each pixel of the smears. In all cases, the amount of excess amplitude added to each target pixel was equal to  $3\sigma_e$ , where  $\sigma_e$  was calculated using Equations 3.1.2 and 3.1.3.

Each of the four prefilter algorithms was invoked upon each of the 30 input frames. Hence, 120 frames were prefiltered in all. To account for finite word length effects, all prefiltering calculations were carried out using integer arithmetic with 13-bit two's complement integers. After each prefiltering operation, Equation 3.1.1 was used to calculate SCE, and Equation 3.1.4 was used to calculate  $\beta$ . These calculations were carried out using floating-point arithmetic. For each algorithm,  $\beta$  and SCE were averaged over the 10 input frames containing one-pixel targets, over the 10 input frames containing two-pixel target smears, and over the 10 input frames containing four-pixel target smears. The results are discussed in Section 3.3.2. For each type of target, the average SCE of each algorithm is shown in Table 2. The beta factors are shown in Table 3.

**Table 2. Average SCE data for the experiment of Section 3.3**

	Target Smear Size		
	<u>1-Pixel</u>	<u>2-Pixel</u>	<u>4-Pixel</u>
Point Detection Filter.....	1.7547	1.6668	1.5472
RA I.....	1.9777	1.9453	1.8809
RA II.....	1.7855	1.7503	1.6947
RA III.....	1.4532	1.3621	1.3476

Table 3. Average beta factors for the experiment of Section 3.3

	Target Smear Size		
	<u>1-Pixel</u>	<u>2-Pixel</u>	<u>4-Pixel</u>
Point Detection Filter.....	0.7580	0.7262	0.6810
RA I.....	0.7811	0.7700	0.7526
RA II.....	0.7621	0.7500	0.7321
RA III.....	0.6557	0.6162	0.6117

### 3.3.2 Experimental Results

The average unfiltered clutter standard deviation of the 10 frames containing one-pixel targets was 1297.720 units. For the 10 frames containing two-pixel target smears, the average was 1298.560 units. Finally, the average for the 10 frames containing four-pixel target smears was 1298.483 units.

The point detection filter had an average beta factor of  $7.580 \times 10^{-1}$  and an average SCE of 1.755 for the one-pixel targets. For the two-pixel target smears, the average beta factor was  $7.262 \times 10^{-1}$  and the average SCE was 1.667. Finally, for the four-pixel target smears, the point detection filter had an average beta factor of  $6.810 \times 10^{-1}$  and an average SCE of 1.545. As expected, this filter did not perform as well with multi-pixel target smears as it did with true point targets. Averaging the point detection filter beta factors for each target type yields a result approximately equal to the beta factor that was observed for this filter in Section 3.2, where both one-pixel targets and four-pixel target smears were simultaneously present in the input. This is also true for the signal to clutter ratio enhancement. Apparently, realistic clutter was no more difficult for the point detection filter to discriminate against than was the monotonic clutter used for the experiments of Section 3.2.

For one-pixel targets,  $RA$ , had an average beta factor of  $7.811 \times 10^{-1}$  and an average SCE of 1.978. For two-pixel targets smears, the average beta factor was  $7.700 \times 10^{-1}$  and the average SCE was 1.945. Finally,  $RA$ , had an average beta factor of  $7.526 \times 10^{-1}$  and an average SCE of 1.888 for the four-pixel target smears. The signal to clutter enhancement was approximately five times greater when this filter was run against monotonic clutter in the experiment of Section 3.2. The beta factors for the realistic

imagery were only about 20 percent lower than the beta factor observed for  $RA_I$  in Section 3.2, however. These results were expected, as the realistic imagery contained random noise and irregular clutter features. Consequently, the output clutter variance was substantially greater for the realistic input imagery than for the monotonic imagery used in Section 3.2. As was pointed out in Section 3.1, signal to clutter enhancement is a stronger function of clutter rejection than is the beta factor. The performance of  $RA_I$  was better than that of the point detection filter, significantly so for blurred targets. The presence of multi-pixel target smears in the input did not degrade the performance of  $RA_I$  as much as it did the performance of the point detection filter.

$RA_{II}$  had an average beta factor of  $7.621 \times 10^{-1}$  and an average SCE of 1.786 for the one-pixel targets. For the two-pixel target smears, the average beta factor was  $7.498 \times 10^{-1}$  and the average SCE was 1.750. For the four-pixel target smears,  $RA_{II}$  had an average beta factor of  $7.321 \times 10^{-1}$  and an average SCE of 1.695. The performance of  $RA_{II}$  was quite similar to that of  $RA_I$ , with  $RA_{II}$  being evaluated slightly lower by both criteria. The superiority of  $RA_I$  was more marked in the signal to clutter enhancement than in the beta factor. From this result we conclude that the output clutter variance of  $RA_{II}$  is more sensitive to random noise in the input than is the output clutter variance of  $RA_I$ . Due to the compensation in the  $RA_{II}$  masking operator, the beta factor of this algorithm was slightly less sensitive to the number of pixels in the targets than was the beta factor of  $RA_I$ .

The average beta factor of  $RA_{III}$  was  $6.557 \times 10^{-1}$  for the one-pixel targets, while the average SCE was 1.453. For the two-pixel target smears,  $RA_{III}$  had an average beta factor of  $6.162 \times 10^{-1}$  and an average SCE of 1.362. Finally, for the four-pixel target smears  $RA_{III}$  had an average beta factor of  $6.117 \times 10^{-1}$  and an average SCE of 1.348.

Of the four algorithms evaluated on realistic imagery,  $RA_{III}$  rated poorest in all respects. The output of the five-point median filter mask was too smooth in the horizontal direction, and consequently this algorithm suffered from substantial clutter leakage.

## 4 Real Time Sorting for Algorithm Variant II

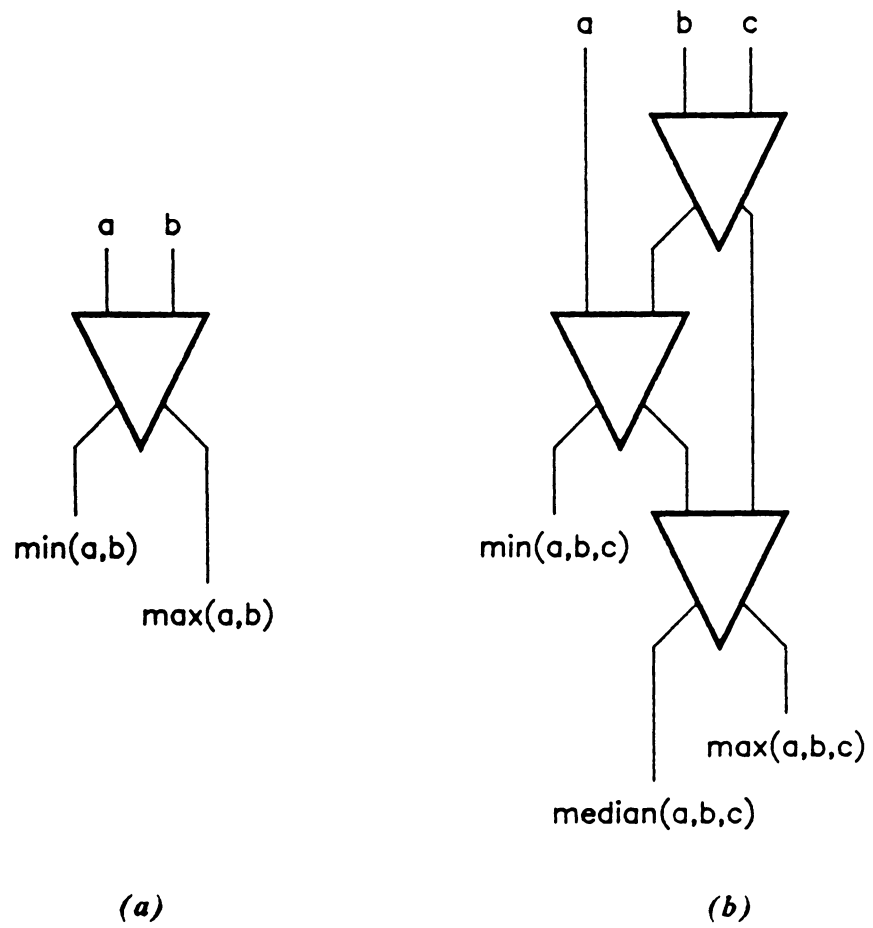
The most difficult problem in median filter implementation is the design of a mechanism for sorting the pixels in the filter window. Since  $RA_I$  and  $RA_{III}$  employ conventional median filters as masking operators, this chapter will concentrate on the development of an architecture to perform the sorting operation required for the implementation of the compensated median filter used in  $RA_{II}$ . This median filter is unique in that the number of pixels to be sorted is even. Fast software sorting algorithms, such as the quicksort which has time complexity  $O(N)$ , are generally used to extract the medians for low data rate median filtering. Even the fastest software algorithms are too slow for use in high data rate applications such as the ATWS prefilter, however. For this system one filtered output must be produced every 509 nanoseconds, which is about equal to the time required to execute one or two instructions on most general purpose microcomputers. In the open literature, three methods have been proposed for the implementation of real time sorting for median filters. Each is briefly described in this chapter. A sorting architecture for  $RA_{II}$  is presented in Section 4.4.



## 4.1 Selection Networks

The most obvious hardware approach to sorting the pixels in the filter window is to employ an array of comparators called a *selection network*. The icon that will be used to represent hardware comparators in this thesis is shown in Figure 15(a). The samples  $a$  and  $b$  at the top of the comparator are input pixels. The bottom left output is the smaller of  $a$  and  $b$ , while the bottom right output is the larger of the two. A simple selection network to find the median of three inputs is shown in Figure 15(b). Note that this network sorts the three inputs in a single machine cycle, and that extraction of the median does not in general imply that the samples in the window must be fully sorted. Selection networks are attractive because they have constant time complexity. Due to the propagation delay of hardware comparators, the sorting delay associated with a digital selection network is in practice a strong function of the number of bits, or *word length*, used to represent pixels. Provided that the comparator propagation delays are short enough for a comparison operation to be considered as a single operation, the time complexity of the selection network method is not a function of the number of pixels in the filter window. The disadvantage of selection networks is that they generally have a greater hardware complexity than other methods.

Adhering to the convention established in Chapter Two, we let  $N$  denote the number of pixels in the filter window. We let  $\Gamma(N)$  denote the minimum number of comparators required in a selection network that extracts the median of  $N$  inputs.  $\Gamma(N)$  is not known for large values of  $N$ , and  $N$  is usually large for image processing applications [34]. Using a proof of Alekseyev [31], Knuth [30] has shown that  $\Gamma(N)$  is bounded by:



**Figure 15. Selection networks:** (a) single comparator; (b) network to extract the median of three pixels.

---

$$\Gamma(N) \geq \left\lceil \frac{N-1}{2} \right\rceil \log_2 \frac{N+3}{2} + \log_2 \frac{N+1}{2} \quad (4.1.1)$$

Ataman, Aatre, and Wong have concluded that selection networks become too complex for practical implementation when  $N > 5$  [34]. Narendra asserted that  $\Gamma(9) = 29$  and  $\Gamma(25) = 94$  [35]. In an effort to reduce the hardware complexity, Shamos [27] and Eversole, *et al.* [28], have implemented approximate median filters using digital selection networks. The approximation involves partitioning the pixels in the window into  $k$  small groups. The median of each group is found, yielding  $Y = \{v_1, \dots, v_k\}$ . The true median is then estimated by  $\hat{v} = \text{MF}[Y]$  [32]. This method is advantageous because the hardware complexity of selection networks is a steeper-than-linear function of the number of inputs. Hence,  $k + 1$  small networks are less complex than a single network, even though the total number of inputs is the same in either case. As was mentioned in Section 2.3.3, Narendra has suggested the implementation of separable two-dimensional median filters by applying one-dimensional median filters independently to the rows and columns of frames. To achieve real time video rates, he specifically recommended use of the analog diode selection networks proposed by Morgan [33]. The advantage of this method is that two  $N$ -point median filters are less complex than a single  $N \times N$  median filter.

## 4.2 Histogram Method

Huang, Yang, and Tang [22], and Garibotto and Lambarelli [19] independently developed a fast method for extracting the median that is based on the histogram of the

pixels in the filter window. We define the histogram as the function  $h(\cdot)$ , where  $h(a)$  is equal to the number of pixels with amplitude  $a$  that are present in the window. We then define a cumulative frequency function by:

$$\lambda(k) = \sum_{a=0}^k h(a) \quad (4.2.1)$$

The median is the smallest value  $v$  with  $h(v) \neq 0$  that satisfies the condition:

$$\lambda(v) \geq \frac{N+1}{2} \quad (4.2.2)$$

The sequence  $Y = \{v_{i,j}\}$  of running medians required for median filtering can be computed quickly using this method. Suppose that an  $M \times M$  square window is being used, and that  $v_{i,j}$  is known. When the window is displaced by one pixel for the computation of  $v_{i,j+1}$ , the histogram  $h(\cdot)$  is updated. In the worst case, all of the  $M$  pixels entering the window have different values from the  $M$  pixels leaving the window. Consequently, updating the histogram may require as many as  $M$  subtractions and  $M$  additions. Computing  $\lambda(\cdot)$  from the updated histogram, if  $\lambda(v_{i,j}) > [(N+1)/2]$ , then  $v_{i,j+1}$  is found by decrementing  $v_{i,j}$  to the smallest value for which Equation 4.2.2 is satisfied and  $h(v_{i,j+1}) \neq 0$ . If  $\lambda(v_{i,j}) < [(N+1)/2]$ , then  $v_{i,j+1}$  is found by incrementing  $v_{i,j}$  until the same conditions are satisfied.

There are two disadvantages to using the histogram method. First, the hardware decision logic required to update the histogram and compute the median is quite complex, especially at the beginnings and ends of rows. Secondly, the number of "bins" required for the histogram is an exponential function of the word length. Consequently, the worst case time complexity also depends exponentially on the word length.

### 4.3 Radix Method

Ataman, Aatre, and Wong proposed the radix method as a fast algorithm for extracting the median of the elements in the filter window [34]. In this method, the  $k^{\text{th}}$  bit of the median is deduced from the  $k$  most significant bits of the pixels in the window. Once again employing the notation of Chapter Two, we let  $W$  represent the set of pixels in the window and  $N$  represent the cardinality of this set. The first bit of the median is found by examining the first bits of all pixels in  $W$ . If a majority of these are ones, then the first bit of the median is a one. If a majority are zeros, then the first bit of the median must be a zero.  $W$  is then partitioned into the set  $S$  of pixels whose first bit is equal to that of the median, and the set  $D$  of pixels whose first bit is different from that of the median. The median is obviously a member of  $S$ , and if  $D$  is non-empty then it is not the median of  $S$ . However, which order statistic of  $S$  is the median of  $W$  can be determined from the cardinalities of  $S$  and  $D$ . The second bit of the median is determined from the second bits of the elements of  $S$ , and  $S$  is subsequently divided into those elements whose second bits equal that of the median and those elements whose second bits differ from that of the median. The procedure continues recursively until all bits of the median have been determined. The cardinality of each partition is saved through the construction of a tree-like data structure in a dedicated register set, or in a random access memory if  $N$  is large. Provided that dedicated parallel hardware is constructed to determine the majority function of the  $k^{\text{th}}$  bits among the elements of  $S$ , the time complexity of the radix method is a linear function of the word length and does not depend on  $N$ .

The disadvantages of using the radix method are that extremely complex hardware decision logic must be implemented to partition the window elements, and that in practice the time complexity may be degraded by the significant number of operations required to maintain the tree structures and deduce the medians. If the word length is less than  $\lceil(N+1)/2\rceil$  bits, then the radix method can be modified to make the computation of running medians slightly faster than the computation of individual medians by retaining some of the partitions when the window is moved [34].

Delman implemented a  $5 \times 5$  median filter with a throughput of 10 million eight-bit pixels per second by simplifying the decision logic required for the radix method [36]. His method, which we will call the *modified radix method*, eliminates the need for computing and saving the cardinality of each partition by recoding the bits of the elements of  $W$ . Delman's example of using the modified radix method to find the median of five three-bit pixels is shown in Figure 16 [36]. In the original data, the first bit of each pixel is examined. Since a majority of these are zero, the first bit of the median is zero. All bits to the right of the first are then changed to ones for all pixels whose first bit is not a zero. In the second step, the second bits of the recoded pixels are examined. Since a majority of these are ones, the second bit of the median must be a one. The third bit is then changed to zero for all pixels whose second bit is zero (ie., not equal to that of the median). Finally, since a majority of the third bits of the recoded pixels are zeros, the third bit of the median must be zero. Assuming that the majority function and recoding operation can both be performed in a single operation, the time complexity of the modified radix method is equal to the word length used to represent pixels. Pipelined hardware could be designed to produce a median every machine cycle.

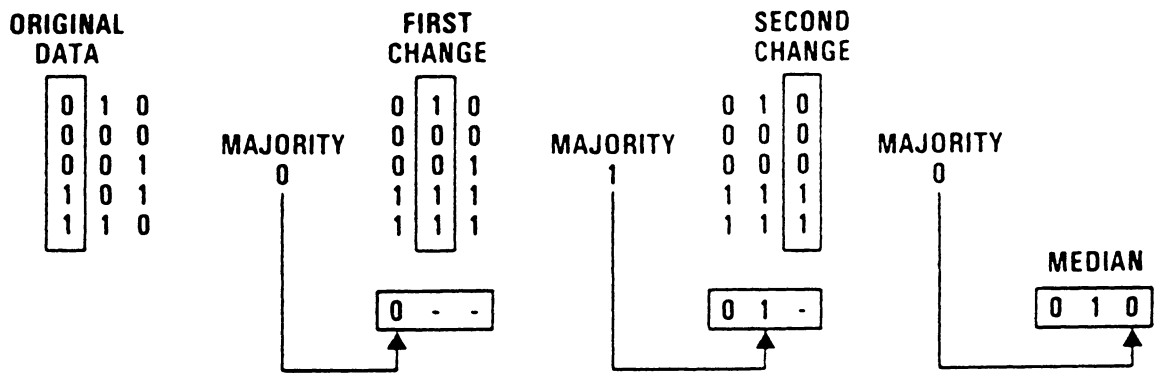


Figure 16. Modified radix method (from reference [36])

## 4.4 Selection Network with In-Place Computation

We now address the problem of developing a sorting architecture for  $RA_{II}$ , and note that for this algorithm variant  $N = 8$ . The word length in the ATWS is 12 bits. Although Equation 4.1.1 tells us that  $\Gamma(8) \geq 11$ , the actual number of comparators required to implement a selection network for  $RA_{II}$  is 24. The selection network is shown in Figure 17, where  $a$  and  $b$  are the two middle values from among the inputs. As only a partial sort is effected by this network,  $a$  and  $b$  do not appear in any particular order. To emphasize the symmetry of the selection network, four additional unnecessary comparators are included in Figure 17. These four are each marked with the character  $X$ . The standard integrated circuit used to implement comparators is the 74n85, where  $n$  is a character representing a particular fabrication technology. This device compares two four-bit integers, and can be cascaded to construct larger comparators of the type required for the selection network of Figure 17. In total, a selection network for  $RA_{II}$  would require the incorporation of 72 74n85 devices. We consider this number too large to be of practical interest.

The histogram method requires maintenance of a number of histogram bins equal to the number of values that can be assumed by a pixel. For the ATWS, this number is  $2^{12} = 4096$ . Although a contents-addressable random access memory could be used to compute the histogram rapidly for  $RA_{II}$ , in the worst case we might have to perform 4095 comparison operations to find the median of  $W$ . Hence we conclude that the histogram method cannot be used to implement a sorting architecture for this application.



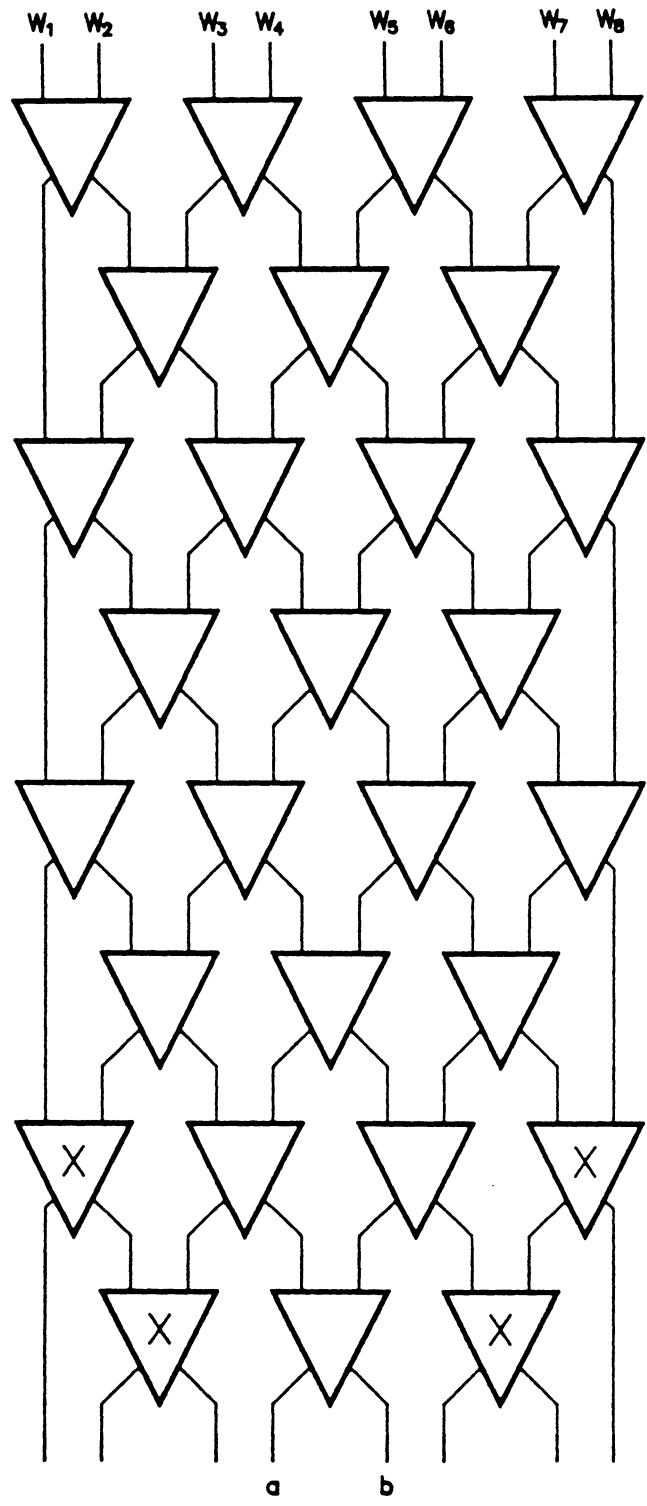


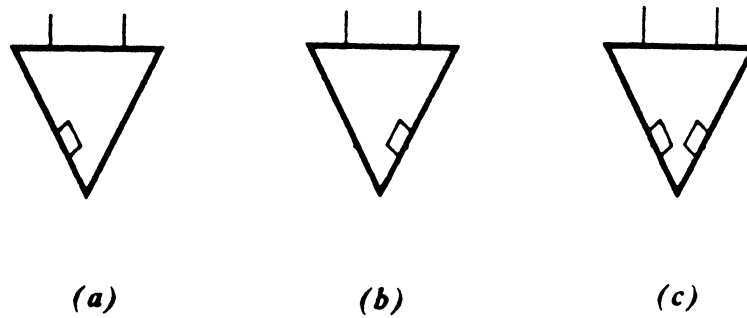
Figure 17. Network to extract the median of eight inputs

Due to the complex decision points inherent in the algorithm, the author knows of no case in which the radix method of Ataman, Aatre, and Wong has actually been implemented in hardware. The modified radix method of Delman would be an attractive alternative, except for the fact that it too involves complex decision logic when it is extended to include the case of  $N$  even. We conclude that the radix method is too complicated to be of practical interest for  $RA_{II}$ .

Returning to Figure 17, we note that if this network were implemented it would be extremely fast and require only one machine cycle to extract the middle two values from the inputs. Since the comparator array is highly symmetrical, we reason that we could implement only two rows of it in hardware, and then reuse these two rows four times to extract the middle values. Such hardware reuse is generally referred to as *in-place computation*. The possibility for such a simplification in the implementation of  $RA_{II}$  was first observed by Mr. K.A. Sarkady of Sachs/Freeman Associates Inc., Landover, MD, who subsequently suggested it to the author.

Repeated use of two rows of the selection network requires the incorporation of memory into the comparator units. Figure 18(a) shows a comparator with a storage register to retain the smaller of its two inputs, while the comparator in Figure 18(b) remembers the larger of its inputs. The comparator shown in Figure 18(c) has registers for storing both of its outputs. A practical architecture capable of performing the sort operation required for  $RA_{II}$  is shown in Figure 19. On the first iteration through the comparator network, the multiplexers at the top of the figure route the eight pixels of  $W$  to the first row of comparators. On all subsequent iterations, these multiplexers select the pixels being fed back from the comparator network. After four iterations, this in-place selection network extracts the two middle values  $a$  and  $b$  from  $W$ , just as the selection network of Figure

17 would do in a single machine cycle. Implementation of the architecture of Figure 19 requires only 21 74n85 devices, however. We note that a four-fold increase in time complexity has been traded for a factor of 3.4 reduction in hardware complexity. In Chapter Five, the sorting architecture of Figure 19 will be incorporated into a complete real time hardware implementation of  $RA_{II}$ . We will choose the system clock rate fast enough to permit the expenditure of five clock cycles in computing each neighborhood median.



**Figure 18. Comparators with memory:** (a) comparator that remembers the smaller input; (b) comparator that remembers the larger input; (c) comparator that remembers both inputs.

---

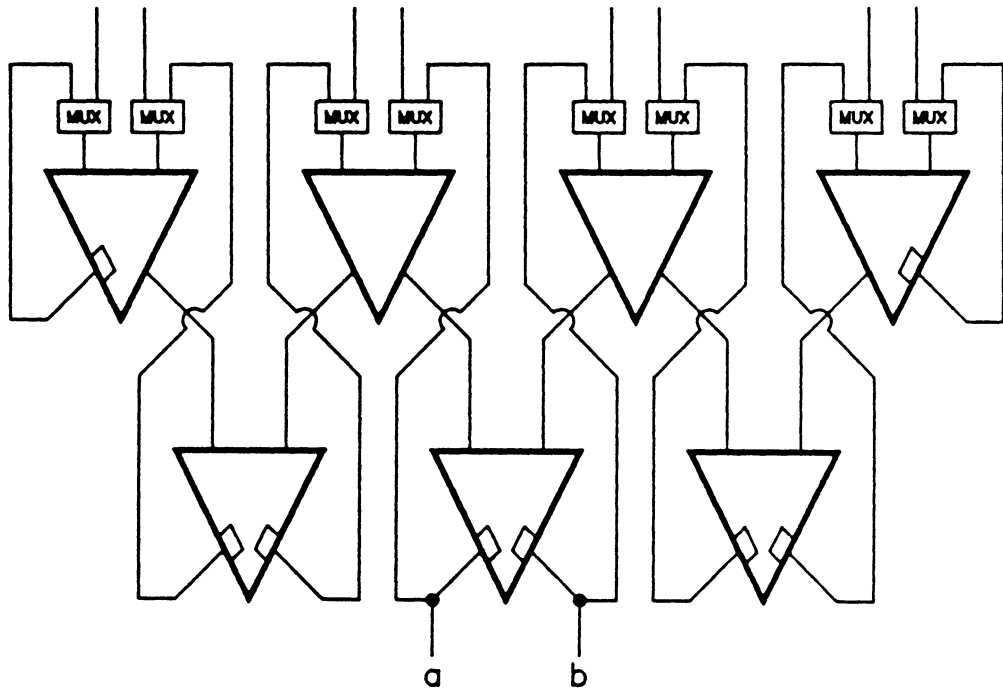


Figure 19. Sorting architecture for algorithm variant II

---

## 5 A Complete Real Time Design

In this chapter, a complete real time hardware implementation of  $RA_{II}$  is presented for the ATWS. A schematic diagram of the design is included in appendix A. Individual drawings of the schematic are referenced by sheet number. The primary reason for choosing  $RA_{II}$  for implementation in this thesis is that, because  $N$  is even for this variant of the recommended algorithm, implementation of the median filter masking operator for  $RA_{II}$  requires the special non-standard sorting architecture developed in Section 4.4. With some modification, the design could also be used to realize real time implementations of  $RA_I$  and  $RA_{III}$ . In particular, since these algorithms employ median filters for which  $N$  is odd, one could modify the design by replacing the sorting architecture of Section 4.4 with an architecture based on the modified radix method of Delman. The design presented in this chapter was verified by a chip-level simulation using the HILO circuit simulation language under the Unix operating system on an HP 9000 minicomputer. The control microcode was further independently verified by a minimal simulation of the design control states. The latter simulation was coded in the REXX language under the CMS operating system on an IBM 3090 computer system. We begin the presentation of the design with the problem specification.

## 5.1 Design Specification

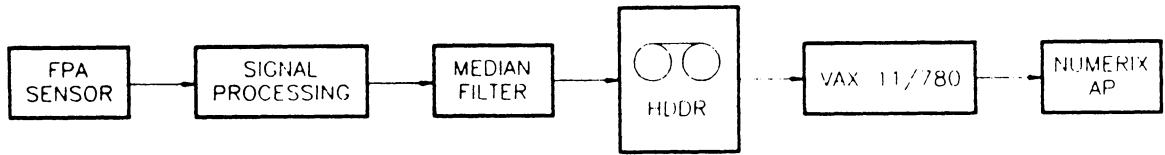
Figure 20 shows simplified block diagrams for two possible prototype configurations of the ATWS. The data rate of the focal plane array sensor may be up to 120 frames per second. The frame size may be  $128 \times 128$  pixels or  $64 \times 64$  pixels. Each pixel output by the sensor is represented by a 12-bit unsigned binary integer data word with an optional 13<sup>th</sup> sign bit, which is required in the system data path because some output pixels from the recommended algorithm may be negative. Hence the system throughput may be as high as 25.559 million bits per second. As they are still in a developmental stage, the ATWS target recognition, classification, and tracking algorithms are implemented in software on a VAX 11/780 computer and Numerix array processor. Obviously, these machines are not capable of keeping pace with the sensor data rate. Consequently, both of the system configurations shown in Figure 20 involve a *high density data recorder*, or HDDR. This device is capable of recording several seconds of imagery to tape at the sensor data rate. In the configuration of Figure 20(a),  $RA_{II}$  is implemented in the box labeled MEDIAN FILTER, and consequently must operate at the sensor data rate. Filtered data are recorded on the HDDR, and later read back at a rate slow enough for processing on the VAX 11/780 and array processor. In the configuration of Figure 20(b), unfiltered data are recorded on the HDDR.  $RA_{II}$  is installed after the HDDR, and consequently must operate at the lower data rate. We conclude that the implemented version of  $RA_{II}$  must be capable of operation under at least two widely different data rates. Furthermore, the prefilter must be transparent in the sense that the signals at its output must be indistinguishable from the signals at its input. This requirement arises from the fact that the HDDR must be conveniently

interfaced to either the prefilter output or the post-sensor signal processing assembly, depending on which system configuration is being used at any given time.

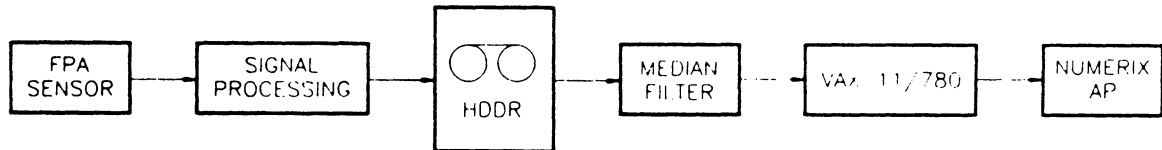
The timing characteristics of the system data path are shown in Figure 21. The signal CLOCK is a twisted-pair strobe. CLOCK is normally high, and all other signals are valid on its falling edge. The only constraint that applies to CLOCK is that consecutive falling edges must be separated by a time interval of at least 509 nanoseconds, corresponding to a maximum data rate of  $120 \times 128 \times 128$  frames per second. In particular, CLOCK is not required to have a regular duty cycle, and it is not required to be periodic. The signal FRAMESYNC is a scalar twisted-pair strobe that is stable at high during all falling edges of CLOCK that correspond to the first pixel of a frame. Any spurious number of falling edges may occur on CLOCK between the last pixel of one frame and the first pixel of the next frame. The beginning of the new frame is recognized only when FRAMESYNC is high. The signal vector DATA comprises 12 twisted-pair lines for the pixel data and an optional 13<sup>th</sup> twisted-pair sign bit. The vector DATA is valid on all falling edges of CLOCK, except those that occur between the end of one frame and the beginning of the next.

Two peculiarities apply to the signals at the output of the  $RA_{II}$  prefilter. First, the sense of CLOCK is reversed from the convention stated above. That is, the prefilter output signal corresponding to the input signal CLOCK is actually  $\overline{\text{CLOCK}}$ , and all other signals are valid on the rising edge of  $\overline{\text{CLOCK}}$ . This convention was mandated by the proprietors of the ATWS. With respect to the two configurations of Figure 20, this does not pose a system integration problem since the sense of any twisted-pair signal can easily be reversed by simply cross wiring a special interconnect cable. Secondly, the vector DATA at the prefilter output is interpreted as a 13-bit two's complement integer.





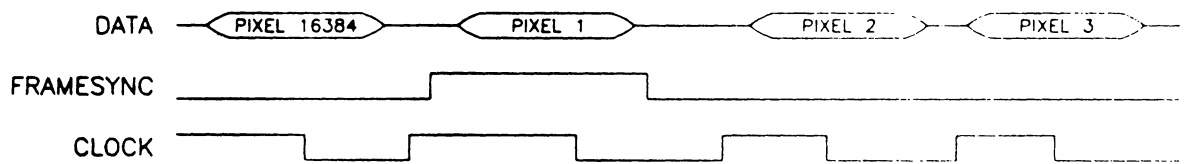
(a)



(b)

**Figure 20. Simplified ATWS system block diagram:** (a) system for recording prefiltered data; (b) system for prefiltering recorded data.

---



**Figure 21. System data path timing characteristics for the ATWS**

---

At the prefilter input, the sign bit has no significance and the pixels are assumed to be represented by non-negative 12-bit integers. As stated above, the sign bit in the prefilter output is necessary because, even though all input pixels are non-negative,  $RA_{II}$  may produce output pixels that are positive or negative due to the subtraction operation inherent in the algorithm.

In the practical implementation of  $RA_{II}$ , several complications arise that were not present when the algorithm was evaluated in Chapter Three. Here, we present these complications by listing a set of rules for producing filtered output. This list will be referred to as the *design specification list* in Section 5.2. The rules are:

1. The data in the first row of any input frame is header information, and does not represent pixels. The header row must be transmitted to the prefilter output without alteration.
2. Since the first row of any frame consists header information, the frame actually contains only 127 or 63 rows of pixel data, with the first of these being the second row of the frame. As the compensated median filter window is of size  $3 \times 3$  pixels, this results in the second and last rows of any output being indeterminate. Corresponding to these indeterminate rows, the prefilter output vector DATA should be tri-stated.
3. Due again to the size of the compensated median filter window, the first and last data words output on any line containing pixel values are indeterminate. Corresponding to such data words, the prefilter output vector DATA should be tri-stated.

4. Detectors in the sensor array sometimes cease to function, or die. Any input pixel with a value of zero is assumed to come from a dead detector. Such zero values must be passed to the prefilter output without alteration.
5. Detectors in the sensor array sometimes become saturated when part of a scene under observation by the ATWS is particularly bright. This is a consequence of the fact that the sensor is a charge integrating device, and only a finite amount of charge can be integrated on the substrate. The prefilter must provide a bank of 12 DIP switches for manually setting a saturation threshold. Any input pixel that is greater or equal in value to the saturation threshold is assumed to come from a saturated detector. Such values must be passed to the prefilter output without alteration.
6. Any input data word not meeting any of the criteria in items one through five above is assumed to come from an active non-saturated detector, and to be interior in the sense that all of its nearest neighbors represent pixels. Such data words are interpreted as pixels. The prefilter output corresponding to these data words is computed using  $RA_{II}$  as specified in Section 2.3.2.

## ***5.2 Design Presentation***

Sheet 1 of the schematic is a block diagram of the entire circuit. For convenience, it is reproduced as Figure 22. In the schematic, low active signals are annotated by postfixing the signal name with a tilde. In the thesis text, low active signals are annotated with an overbar. Circuit module SINGLESTEP generates the system clock,

and circuit module MICRO CTRL is a custom microcode sequencer which generates timing signals and control strobes for the rest of the circuit. Circuit module EXIN SYNC synchronizes the external inputs to the system clock. Pixel data enter the prefilter sequentially in conventional scanning order. Circuit module DELAY LINE is essentially a large shift buffer which holds two full scan lines plus three pixels of data words. The delay line has nine taps, and at any given time a particular pixel  $x_{i,j}$  and its eight nearest neighbors are available at the taps. The sorting architecture presented in Section 4.4 is implemented in circuit module SRT MTRX, and this module also contains additional hardware for the computation of prefilter outputs according to the rules in the design specification list of Section 5.1.

In the present section, the operation of each circuit module will be discussed in some detail. First we describe the circuit data flow at a general level, once again with reference to the block diagram of Figure 22. Data words representing pixel values arrive on the vector EDAT at the top left corner of the figure. Once they are synchronized to the system clock, they are transferred to the delay line on the 12-bit data bus DINBUS. The pixel  $x_{i,j}$  which is at the center of the  $3 \times 3$  filter window is output from the delay line on the data bus XIJBUS. The rest of the pixels in the window are available on the eight data busses XABUS through XHBUS. These nine busses lead to the module SRT MTRX. In this module, the median of the neighborhood is computed and subtracted from  $x_{i,j}$ . The resulting 12-bit data word is output from the prefilter on the data vector DOUT at the lower left of the block diagram. The sign bit is output on the scalar twisted-pair line UNDERFLO. Although components from several logic families appear in the schematic diagram, timing calculations for the circuit were carried out using 74F series components wherever possible. These devices have particularly short rise and fall

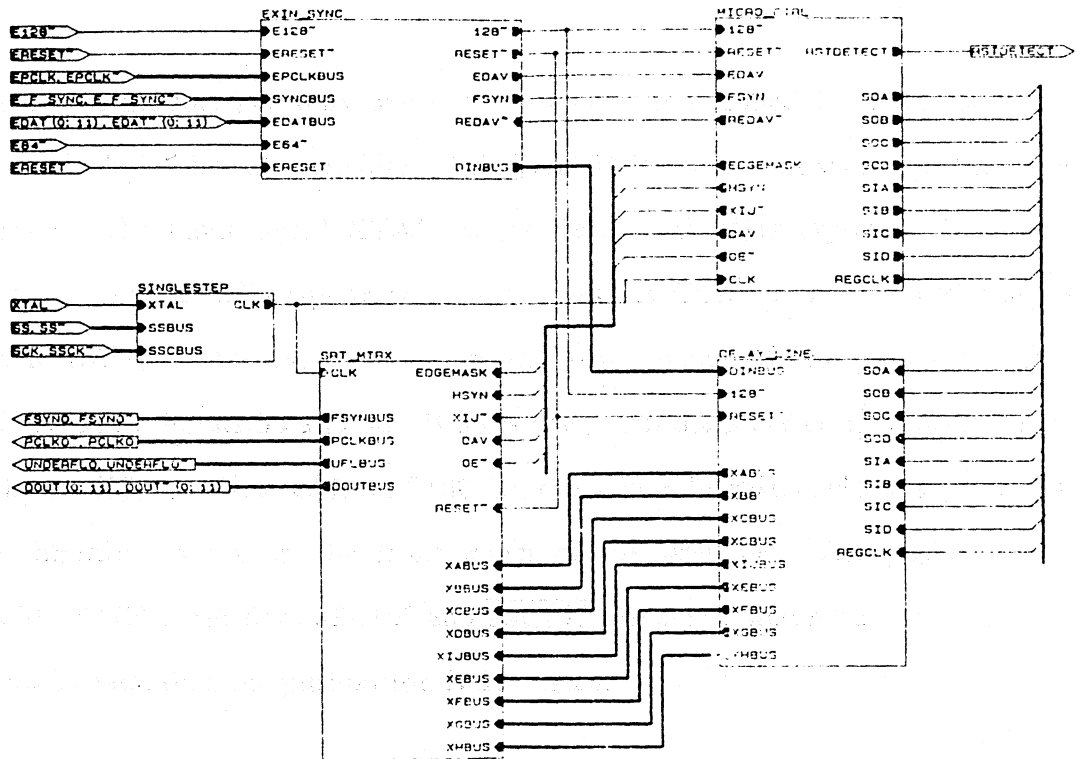


Figure 22. Circuit block diagram

times. We begin our discussion of the circuit modules with SINGLESTEP, which generates the system clock.

### 5.2.1 Circuit Module SINGLESTEP

This circuit module has only one output signal. It is called CLK and serves as the system clock for the entire prefilter. SINGLESTEP appears on sheet 3 of the schematic diagram. The input signal XTAL comes from a 12 MHz crystal. During normal operation, this 12 MHz square wave directly drives CLK. For diagnostic purposes, a bipolar rocker switch is included on the front panel of the prefilter and drives SINGLESTEP inputs SS and  $\overline{SS}$ . If SS is true, then the prefilter operates in single step mode and XTAL is isolated from CLK. In this mode the system clock is generated from a pushbutton switch on the front panel of the prefilter. The pushbutton drives SINGLESTEP input lines SSCLK and  $\overline{SSCLK}$ . In single step mode, one clock pulse is generated each time the pushbutton is depressed.

### 5.2.2 Circuit Module EXIN SYNC

The purpose of this circuit module is to synchronize the prefilter inputs to the system clock. EXIN SYNC appears on sheet 2 of the schematic diagram. This module operates in a totally asynchronous mode, and does not make use of the signal CLK. The names of input signals to module EXIN SYNC are prefixed with the character *E* to indicate that they come from the *external* world.  $\overline{E128}$  and  $\overline{E64}$  are generated from a bipolar switch on the front panel of the prefilter. They drive the inputs of a differential line

receiver in EXIN SYNC to yield a single output signal  $\overline{I28}$ . If  $\overline{I28}$  is low, then the prefilter assumes that the frame size is  $128 \times 128$  pixels. If  $\overline{I28}$  is high, then the prefilter assumes that the frame size is  $64 \times 64$ .

A system reset pushbutton on the front panel of the prefilter drives EXIN SYNC input lines ERESET and  $\overline{ERESET}$ . When this button is depressed, the low active EXIN SYNC output signal  $\overline{RESET}$  goes low and resets the entire prefilter. In addition, a simple RC circuit in module EXIN SYNC holds  $\overline{RESET}$  low for about 300 ms when power is initially applied to the prefilter. Hence, the system always powers up in a reset state.

The remainder of the inputs to EXIN SYNC come from the ATWS data path, which was illustrated in Figure 21. These twisted-pair signals enter the prefilter through a DB-37 connector on the back panel of the chassis. The signal CLOCK of Figure 21 is connected to the EXIN SYNC input bus EPCLKBUS, which stands for *external pixel clock bus*. The negative logic version of this signal is extracted, and will hereafter be referred to as the *pixel clock*. The signal FRAMESYNC of Figure 21 is connected to EXIN SYNC input bus SYNCBUS, while the vector DATA of Figure 22 is connected to EXIN SYNC input bus EDATBUS. On each rising edge of the pixel clock, EXIN SYNC latches the value on SYNCBUS to produce the frame synchronization signal FSYN. FSYN is high for the first pixel of each frame. The value on EDATBUS is latched into the data vector DINBUS, which stands for *data input bus*. Once these values are latched, EXIN SYNC output signal EDAV (which stands for *external data available*) is set high. When the microcode sequencer detects this condition, the value on DINBUS is transferred to the delay line, provided that the pixel clock edge was not a spurious pulse between frames. The microcode sequencer then issues a negative pulse



on the line  $\overline{\text{REDAV}}$  (which stands for *reset external data available*), causing EXIN SYNC to reset and await the arrival of the next rising edge of the pixel clock.

### 5.2.3 Circuit Module DELAY LINE

The delay line appears on sheets 8, 9, and 10 of the schematic diagram. This module does not make direct use of CLK, but rather is driven by control strobes derived from CLK in the microcode sequencer. The delay line taps are a series of nine 12-bit registers which at any time hold the pixels that are currently contained in the  $3 \times 3$  compensated median filter window of  $RA_{II}$ . The outputs of these nine registers drive the nine output busses of the delay line. On each rising edge of the microcode sequencer generated strobe REGCLK, pixel data are advanced one position in the delay line.

As an example, consider that  $x_{i,j}$  is currently in the register which drives XIJBUS. Then  $x_{i+1,j+1}$  is in the register which drives XABUS,  $x_{i+1,j}$  is in the register which drives XBBUS, and  $x_{i+1,j-1}$  is in the register which drives XCBUS. A bank of six 74F413 FIFO buffer devices is connected to the output of the register which drives XCBUS, and all  $x_{p,q}$  which fall between  $x_{i+1,j-1}$  and  $x_{i,j+1}$  in the scan order are stored in these buffers. The delay line input strobes SOA, SIA, SOB, and SIB (which respectively stand for *shift out of buffer A*, *shift into buffer A*, *shift out of buffer B*, and *shift into buffer B*) control the operation of these FIFO buffers. The input  $x_{i,j+1}$  is in the register which drives XDBUS, and the input  $x_{i,j-1}$  is in the register which drives XEBUS. A second bank of six 74F413 FIFO buffers is connected to the output of the register which drives XEBUS, and holds all inputs  $x_{p,q}$  which fall between  $x_{i,j-1}$  and  $x_{i-1,j+1}$  in the scan order. These FIFO buffers are controlled by the delay line input strobes SOC, SIC, SOD, and SID, which are named

according to the same convention as the control strobes for the first bank of buffers. The input  $x_{i-1,j+1}$  is in the register which drives XFBUS,  $x_{i-1,j}$  is in the register which drives XGBUS, and  $x_{i-1,j-1}$  is in the register which drives XHBUS. Hence,  $x_{i,j}$  and its eight nearest neighbors are available at the taps of the delay line. On the next rising edge of REGCLK, the delay line shifts one position. At that time,  $x_{i,j+1}$  enters the register which drives XIJBUS. This pixel and its eight nearest neighbors are then available at the delay line taps. The pixel  $x_{i-1,j-1}$  is shifted out of the delay line and discarded in order to make room for  $x_{i+1,j-2}$  to enter the delay line.

The timing constraints applying to the control inputs of the 74F413 FIFO buffers are quite complex, and will not be discussed here. These devices are each four bits wide and 64 bits deep. When the prefilter operates in  $128 \times 128$  mode, 125 pixels must be stored in each buffer bank. Each bank is constructed from a square array of 74F413 buffers that is three devices wide by two devices deep, collectively providing storage for up to 128 12-bit data words in each bank. In  $64 \times 64$  mode, each bank must store only 61 pixels. Consequently, the registers driving XDBUS and XFBUS were constructed from 74F399 four-bit dual-ported register devices to provide the capability of realizing buffers of either depth. The signal  $\overline{128}$  selects whether these registers latch the pixel at the head of a 61 element queue or at the head of a 125 element queue.

As a final point concerning the delay line, we note that this circuit module must be *primed* by the microcode sequencer after any system reset. By this is meant that two complete scan rows plus three data words must be received at the prefilter input and shifted into the delay line before the first complete neighborhood becomes available at the delay line taps. Furthermore, since the first row of any frame contains only header

information, three complete rows plus three data words must be received before the first filtered output pixel can be produced.

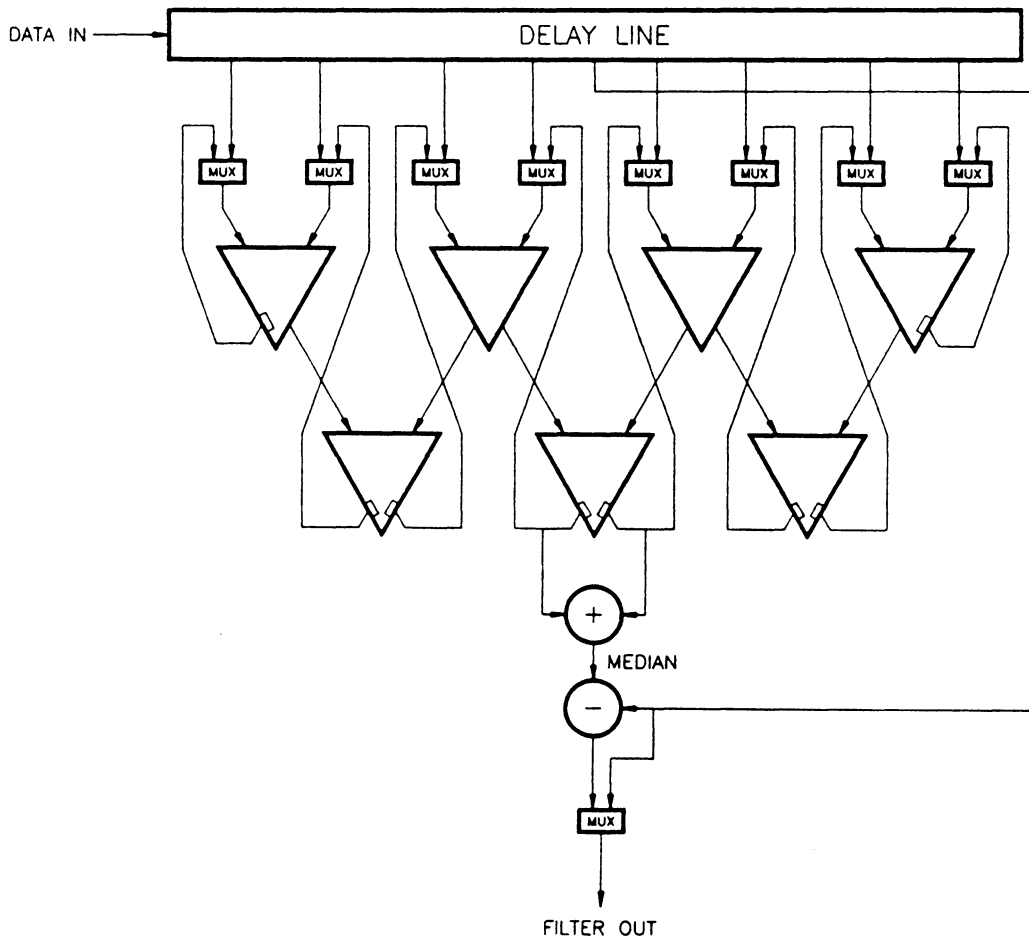
#### 5.2.4 Circuit Module SRT MTRX

The SRT MTRX module appears on sheets 11 through 22 of the schematic diagram. Returning once again to the block diagram of Figure 22, we see that the inputs to SRT MTRX are  $\overline{\text{RESET}}$ , CLK, five control strobes, and the nine data busses output by the delay line. The module outputs are all wired to a DB-37 connector on the back panel of the prefilter, and are all twisted-pair signals. Excepting the inverted sense of the prefilter output clock as described in Section 5.1, the outputs from module SRT MTRX adhere to the ATWS timing characteristics illustrated in Figure 21. 12-bit filtered output pixels are produced on the output data vector DOUT of Figure 22, and correspond to the vector DATA in Figure 21. The scalar output signal UNDERFLO is the output sign bit, which was also discussed in Section 5.1. If this signal is true, then DOUT is interpreted as a negative pixel value. Taking UNDERFLO to be the high order bit of DOUT, one obtains a 13-bit two's complement integer. The output signal FSYNO provides the means of frame synchronization in the filtered output. This signal is true when the first data word of any output frame is stable on the output vector DOUT. Timing characteristics for the signal FSYNO in Figure 22 are the same as for the signal FRAMESYNC in Figure 21. The output pixel clock is PCLKO. DOUT and FSYNO are assumed valid on the rising edge of PCLKO. Since all prefilter outputs are twisted-pair signals taken from differential line drivers, the timing characteristics of signal  $\overline{\text{PCLKO}}$  in Figure 22 are identical to those of the signal CLOCK in Figure 21.

Figure 23 is a block diagram illustrating the computation of prefilter outputs. It is functionally equivalent to the diagram on sheet 11 of the schematics. The data word which is currently at the center of the  $3 \times 3$  compensated median filter mask of  $RA_{II}$  is available on XIJBUS, unless the delay line is being primed. This bus is represented by the center tap of the delay line in Figure 23. If the data word on XIJBUS is a header, then the microcode sequencer asserts the low active signal  $\overline{XIJ}$ , causing the multiplexer at the bottom of Figure 23 to connect XIJBUS directly to the prefilter output vector DOUT. Hence, all headers are passed directly to the prefilter output without alteration as required by item one of the design specification list.

Circuit submodule SATDETECT appears on sheet 22 of the schematic diagram. SATDETECT contains a bank of 12 DIP switches which are used to manually set the saturation threshold. SATDETECT also contains logic which compares the data word on XIJBUS to the saturation threshold and to zero. If the data word on XIJBUS is zero, or if it is greater than or equal to the current saturation threshold, then the output signal SEL from SATDETECT causes the multiplexer at the bottom of Figure 23 to connect XIJBUS directly to DOUT. Hence, as required by items four and five of the design specification list, zero valued pixels and saturated pixels are passed directly to the prefilter output without alteration.

When XIJBUS holds a normal pixel in the sense described by item six of the design specification list, the hardware sorting architecture of Figure 23 operates as described in Section 4.4. During the first cycle of CLK for which this pixel and its eight nearest neighbors are stable on the delay line taps, the microcode sequencer asserts the SRT MTRX input control strobe DAV (which stands for *data available*). During any clock cycle for which DAV is asserted, the multiplexers at the top of the sorting array connect



**Figure 23. Computation of prefiltered outputs**

the delay line taps to the top of the array, allowing a partial sort of the neighborhood to take place. For three subsequent cycles of CLK, the multiplexers feed back the sorting array outputs to the array inputs. Consequently, by the end of the fourth clock cycle the two middle valued pixels of the neighborhood emerge at the bottom of the sorting array. These middle valued pixels are added using a cascade connection of three 74F181 arithmetic logic units, and the low order bit of the sum is dropped to yield the neighborhood median. The median is then subtracted from the pixel on XIJBUS using a second set of three 74F181 arithmetic logic units. The addition and subtraction operations are both performed during the fifth clock cycle after DAV is asserted, and thus  $RA_{II}$  is realized in five cycles of the 12 MHz clock. As the result is a valid filtered output, the multiplexer at the bottom of Figure 23 connects the outputs of the second set of arithmetic logic units to the vector DOUT in this case.

Since computing a filtered output pixel requires five clock cycles, circuit module SRT MTRX contains a simple five phase hardwired control unit to generate internal timing signals. Each control phase lasts for one cycle of CLK. The control unit waits in phase one until DAV is asserted, and then cycles through the other phases. The filtered output is latched into DOUT at the end of control phase five. During the next occurrence of phase one, the output pixel clock PCLKO is generated, provided that the microcode sequencer has not asserted the control signal EDGEMASK. EDGEMASK provides a means of preventing spurious output pulses on PCLKO while the delay line is being primed. Additionally, if the microcode sequencer asserts control signal HSYN, then the current output data word is assumed to be the first header word of a frame. In this case, SRT MTRX asserts the output frame synchronization signal FSYNO. Finally, the control signal  $\overline{OE}$  provides a means for satisfying items two and three of the design

specification list. When the microcode sequencer asserts  $\overline{OE}$ , the data output lines DOUT and UNDERFLO are tri-stated by the SRT MTRX module control unit.

### 5.2.5 Circuit Module MICRO CTRL

Circuit module MICRO CTRL is the prefilter microcode sequencer, and appears on sheets 4 through 7 of the schematic diagram. The architecture for the microcode sequencer is shown in Figure 24. A three phase internal hardwired control unit is employed, and each control phase lasts for one cycle of CLK. During control phase one, the microcode sequencer waits for EXIN SYNC to assert EDAV, signifying that external data has arrived and has been synchronized. When this occurs, the microcode sequencer enters control phase two, during which strobes are asserted to advance the delay line. Control phase two is always followed immediately by control phase three. During phase three, the microcode sequencer asserts the five strobes which control module SRT MTRX, and also sequences to the next microinstruction. Control phase three is always followed immediately by control phase one. Since the microcode sequencer can stay in control phase one for any number of clock cycles, the requirement that the prefilter be able to operate at vastly divergent data rates is satisfied. The internal clock always runs at 12 MHz. In a low data rate application, the microcode sequencer simply spends most of its time in control phase one waiting for input.

The microinstruction format is shown in Figure 25. The implementation of  $RA_{17}$  requires 191 microinstructions, and consequently the microaddress register is eight bits wide. This register is labeled PC in Figure 24, which stands for *program counter*. The microinstructions are 24 bits wide, and the microstore is implemented in three

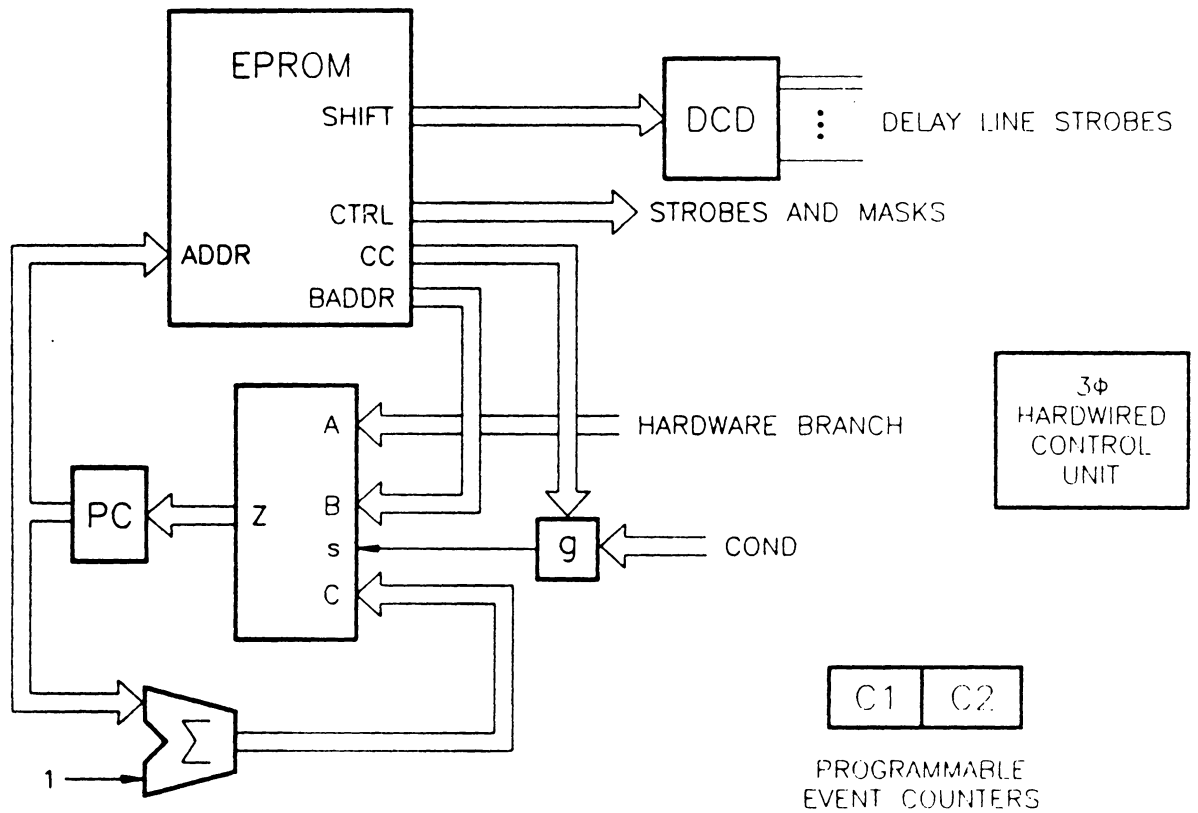


Figure 24. Microcode sequencer architecture



Monolithic Memories high speed  $256 \times 8$ -bit ROMS. The first microinstruction field is called  $P_n$ , and it controls the delay line FIFO buffers. The signal SIA, which was discussed briefly in Section 5.2.3, is coded directly as the first bit of  $P_n$ . The other three bits of  $P_n$  are called P0, P1, and P2. The block labeled DCD in Figure 24 contains combinational logic to decode these bits, and from them generate the FIFO control strobes SIB, SIC, SID, SOA, SOB, SOC, and SOD.

The second microinstruction field is called I, and its four bits are used to generate internal control signals for the microcode sequencer. The first bit of I is called  $WDAV/\overline{HSYN}$ . If this bit is a zero, then the microcode sequencer waits in control phase one until EXIN SYNC receives input data with FSYN active. Stated another way, coding a zero in this bit forces the prefilter to wait for the first pixel of a new frame, irrespective of how many spurious pulses occur on the external data clock. If  $WDAV/\overline{HSYN}$  is coded as a one, then the microcode sequencer waits in control phase one only until EXIN SYNC activates EDAV. The remaining three bits of I are used to control the two internal programmable event counters provided in the microcode sequencer architecture. These counters are labeled C1 and C2 in Figure 24. If the bit LOAD1 is set in the current microinstruction, then C1 is loaded with one of two hardwired values, depending on the value of the operation mode control signal  $\overline{I28}$ . Likewise, if the bit LOAD2 is set, then C2 is loaded with one of two hardwired values. These counters are used by the microcode sequencer to keep track of which pixel of the input frame is currently being processed. C1 counts columns, while C2 counts rows. C1 is decremented every time input is received by EXIN SYNC. If the bit C2E is coded as a one in the current microinstruction, then C2 is also decremented. C2E stands for *counter two enable*.

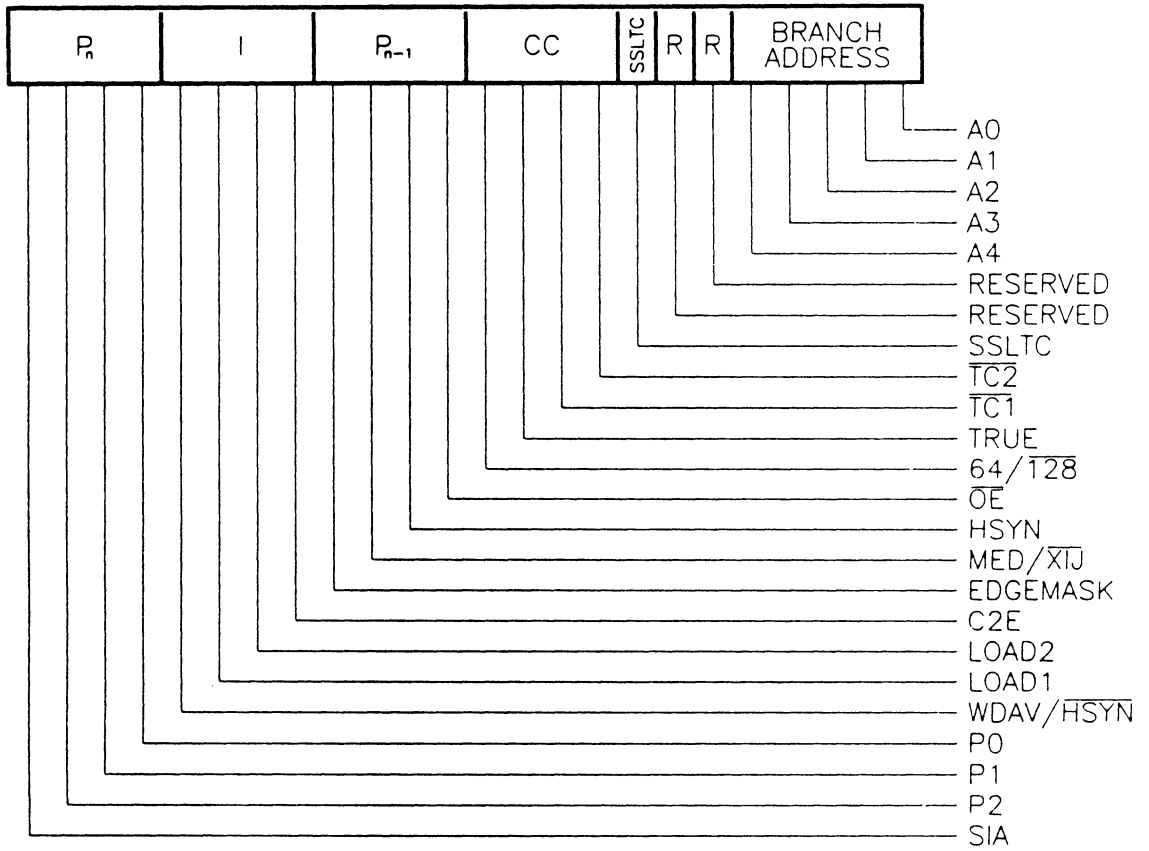


Figure 25. Microinstruction format

The third microinstruction field is called  $P_{n-1}$ , and also contains four bits. This field is used to generate control strobes for module SRT MTRX. The name  $P_{n-1}$  was chosen to emphasize the pipelining between the microcode sequencer and module SRT MTRX. Recall that SRT MTRX employs a five phase internal hardwired controller, and that the prefilter outputs are latched by the output differential line drivers at the end of phase five. Consequently, the prefilter outputs are not actually stable until phase one of the following control cycle. During phase one, however, module SRT MTRX may proceed to fetch a new neighborhood from the delay line and begin sorting it. Whereas the control signals generated from microinstruction field  $P_n$  always pertain to the neighborhood that is currently being transferred from the delay line to module SRT MTRX, the control signals generated from microinstruction field  $P_{n-1}$  determine the characteristics of the prefilter output signals. Consequently, control strobes generated from  $P_{n-1}$  pertain to the prefilter output that was calculated during the previous control cycle of module SRT MTRX.

The four bits of microinstruction field  $P_{n-1}$  are called EDGEMASK,  $MED/\overline{XIJ}$ , and HSYN. They directly generate the signals bearing the same names, with the exception that the name of the signal generated by  $MED/\overline{XIJ}$  is shortened to  $\overline{XIJ}$ . These signals were discussed in Section 5.2.4. DAV, the fifth control strobe generated by the microcode sequencer for use in module SRT MTRX, is always true during microcode sequencer control phase two.

As shown in Figure 24, there are three ways in which the next value of the microaddress register may be determined. Normally, the current microaddress is simply incremented. The microcode sequencer architecture also provides four software branch conditions. The fourth microinstruction field is called CC, and it contains one bit for each of these

branch conditions. CC stands for *condition codes*. Only one bit of CC may be a one in any given microinstruction. During control phase three, combinational logic in the block labelled g in Figure 24 examines the CC field of the current microinstruction. This logic also examines the four corresponding hardware conditions, which are labelled COND in Figure 24. If a bit of CC is set and the corresponding condition is also true, then the logic in g outputs a high value. In this case, the address of the next microinstruction is constructed by replacing the low order five bits of the current microaddress with the five bits of the BADDR field of the current microinstruction. The name BADDR stands for *branch address*. If the output of g is low, then the condition fails and the next address is formed by incrementing the current contents of the microaddress register.

The first bit of the field CC is called  $64/\overline{128}$ . If this bit is set, then a software branch is taken if the current value of mode control signal  $\overline{128}$  is a one. Recall that the signal  $\overline{128}$  is driven by a rocker switch on the front panel of the prefilter. The second bit of CC is called TRUE. If this bit is set, then the microcode sequencer branches unconditionally. The third and fourth bits of CC are called  $\overline{TC1}$  and  $\overline{TC2}$ . If  $\overline{TC1}$  is set, then a software branch is taken unless the terminal count has been reached by counter C1. Likewise, a software branch is also taken if bit  $\overline{TC2}$  is set and counter C2 has not reached its terminal count.

The microcode is arranged in two sections, each of which contain microinstructions to process one frame. The difference between the two sections is that the first, which begins at microaddress zero, contains microinstructions to prime the delay line. Consequently, this section is executed only during processing of the first frame received after the prefilter is reset. The microcode in the second section is repeatedly executed

for all subsequent frames. As was mentioned above, software branching is effected by replacing the five low order bits of the microaddress register with the BADDR field of the current microinstruction. This branching mechanism has the result that the microcode is divided into segments of  $2^5 = 32$  microinstructions each. Software branches can only reach destination addresses within the current segment.

Since the second microcode section is far greater than one segment in length, it is impossible to use a software branch to return to the top of the section. A hardwired branch was designed for this purpose, and represents the third and final method by which the next microaddress may be determined in Figure 24. The parallel load lines of the microaddress register are permanently hardwired with the first address of the second microcode section. The last microinstruction of this section has the bit SSLTC coded as a one, and this bit is coded as a zero in all other microinstructions. After the last microinstruction is executed, the hardwired address is always loaded to the microaddress register. Hence, the microcode sequencer branches back to the top of the section for another iteration. SSLTC stands for *steady state loop terminal condition*. The two microinstruction bits labelled RESERVED in Figure 25 are reserved for use in future modifications.

## 6 Conclusions and Recommendations for Further Research

The ATWS prefilter problem is a difficult one because little is known about the characteristics of backgrounds in midwave infrared imagery. In many cases, the target and clutter spectra may not be disjoint. Since precise models are not currently available for targets or clutter, qualitative design techniques must be employed. The concept of an *ideal* prefiltering algorithm for the ATWS was developed in the opening of Chapter Two. The ideal algorithm passes all targets in the input imagery without attenuation, and it also reduces the values of all non-target pixels to zero. The remainder of the thesis was devoted to the investigation of realizable prefiltering algorithms to approximate the ideal algorithm.

## 6.1 Algorithm Development and Evaluation

In Section 2.1, an image enhancement technique known as unsharp masking was introduced. With this technique, output images are formed by subtracting masked versions of the input images from the originals. The properties of median filters were examined in Section 2.2, and it was hypothesized that a median filter masking operator would be capable of removing targets from the ATWS imagery without significantly distorting the clutter features. Consequently, a prefilter based on unsharp masking with a median filter masking operator would provide a realizable approximation to the ideal algorithm.

Three specific median filter unsharp masking algorithm variants were recommended in Section 2.3.  $RA_I$  employed a  $3 \times 3$  median filter masking operator. It was hypothesized that the presence of multi-pixel target smears and heavily-tailed random noise in the input imagery might bias the amplitudes of the medians computed by this operator upward from the true values of the clutter medians. Consequently, a compensated  $3 \times 3$  median filter masking operator was used in  $RA_{II}$ . Each output pixel of the compensated median filter was computed as the median of the eight nearest neighbors to the corresponding input pixel. This compensation technique was proposed by researchers at the United States Naval Research Laboratory. Finally,  $RA_{III}$  employed a one-dimensional five-point median filter masking operator applied to only the rows of the input frames.

The three recommended algorithm variants were quantitatively evaluated in Chapter Three. For comparative purposes, three linear prefiltering algorithms were evaluated as

well. Algorithms were compared on the basis of signal to clutter ratio enhancement due to prefiltering. Since signal to clutter ratio enhancement favors those algorithms that are best able to reject clutter more than it favors those that are best able to pass targets, a second evaluation metric called the beta factor was proposed in Section 3.1. The beta factor may be loosely interpreted as a percentage measure of how well a given algorithm performs with respect to the ideal algorithm. Which criterion is more important to any particular automatic target recognition system depends on the specific characteristics of the target detection and segmentation algorithms employed by that system.

The experimental results show  $RA_{III}$  to be an inferior prefiltering algorithm for the ATWS.  $RA_I$  and  $RA_{II}$  were consistently superior to  $RA_{III}$ , and in some cases the latter did not perform as well as the linear point detection filter. For the input imagery that was considered, a better estimate of the clutter was obtained from a  $3 \times 3$  square window than from a one-dimensional five-point window. As  $RA_{III}$  is more economically implemented in hardware than are  $RA_I$  and  $RA_{II}$ , implementation of  $RA_{III}$  might be considered as an alternative to a  $5 \times 5$  two-dimensional filter for some applications.

The clutter considered in Section 3.2 was smooth and noise free. With the exception of the square-containing input frame of Figure 7, it was also locally monotonic. In the experimental results,  $RA_I$  and  $RA_{II}$  were shown to perform nearly as well as the ideal algorithm. They were both consistently superior to the linear point detection filter.  $RA_I$  enhanced the signal to clutter ratio approximately 20 percent more than  $RA_{II}$ . This was a consequence of slightly greater clutter leakage in the output of the latter.  $RA_{II}$  passed slightly more target energy than  $RA_I$ , and consequently had a slightly greater beta factor. The Laplacian filter and unsharp masking with a  $3 \times 3$  mean filter mask were both inferior to the other algorithms.



Realistic input imagery was considered in Section 3.3. Due to random noise and irregular clutter features, the performances of the recommended algorithm variants were significantly degraded from that observed in Section 3.2.  $RA_I$  and  $RA_{II}$  were consistently superior to the linear point detection filter. All three of these algorithms performed comparably against imagery containing true point targets. In the presence of multi-pixel target smears, however, the performance of the point detection filter was degraded more severely than that of the recommended algorithm variants. As discussed in Section 3.2, this result was expected. For all three target types, the signal to clutter ratio enhancement due to  $RA_I$  was approximately 11 percent greater than that due to  $RA_{II}$ . The beta factor of  $RA_I$  was approximately 2.6 percent greater than that of  $RA_{II}$ , irrespective of the target type. These results indicate that  $RA_I$  and  $RA_{II}$  were approximately equal in their ability to pass the targets, while the filtered clutter variance was somewhat lower for  $RA_I$ . Stated another way, for the experimental input imagery the compensation in the masking operator of  $RA_{II}$  provided no tangible advantage over the uncompensated median filter. One explanation for this might be that the clutter distribution did not have large enough tails for  $RA_{II}$  to excel. The author recommends extensive further investigation of  $RA_I$  and  $RA_{II}$  before either one of these is declared superior to the other. In particular, both algorithms should be evaluated on a large number of realistic frames. The input for the experiments of Section 3.3 contained only 10 unique background scenes.

With respect to the algorithm measurement criteria used in this thesis, it is probable that the performance of both  $RA_I$  and  $RA_{II}$  could be significantly improved if these prefilters were followed by a simple thresholding operation. The filtered clutter mean is expected to be nearly zero, and a good deal of the clutter leakage around targets is expected to have negative amplitude. These assertions are corroborated by the output frames shown

in Section 3.2. Although thresholding the output of  $RA_I$  and  $RA_{II}$  would most certainly reduce the clutter standard deviation, it is questionable whether such an operation would actually facilitate target detection.

## ***6.2 Algorithm Implementation***

Because the determination of local order statistics is a highly nonlinear operation, the design of real time median filters is difficult. The histogram and radix methods resulted from various authors efforts to circumvent the hardware complexity associated with full scale selection networks. Unfortunately, these methods both require extremely complex control logic. In Section 4.4, in-place computation was used to simplify the hardware complexity of the standard selection network. A practical real time sorting architecture for  $RA_{II}$  was developed. The author recommends that this architecture be further investigated with the objective of applying it to the implementation of real time median and order statistic filters in general. That  $RA_I$  and  $RA_{II}$  are in fact *realizable* approximations to the ideal algorithm was demonstrated by the complete design presented in Chapter Five. Although only  $RA_{II}$  was implemented, with reasonable effort the design could be modified to realize  $RA_I$ . In particular, the author would recommend that the sorting architecture developed in Section 4.4 be replaced with an architecture based on the modified radix method.

## Literature Cited

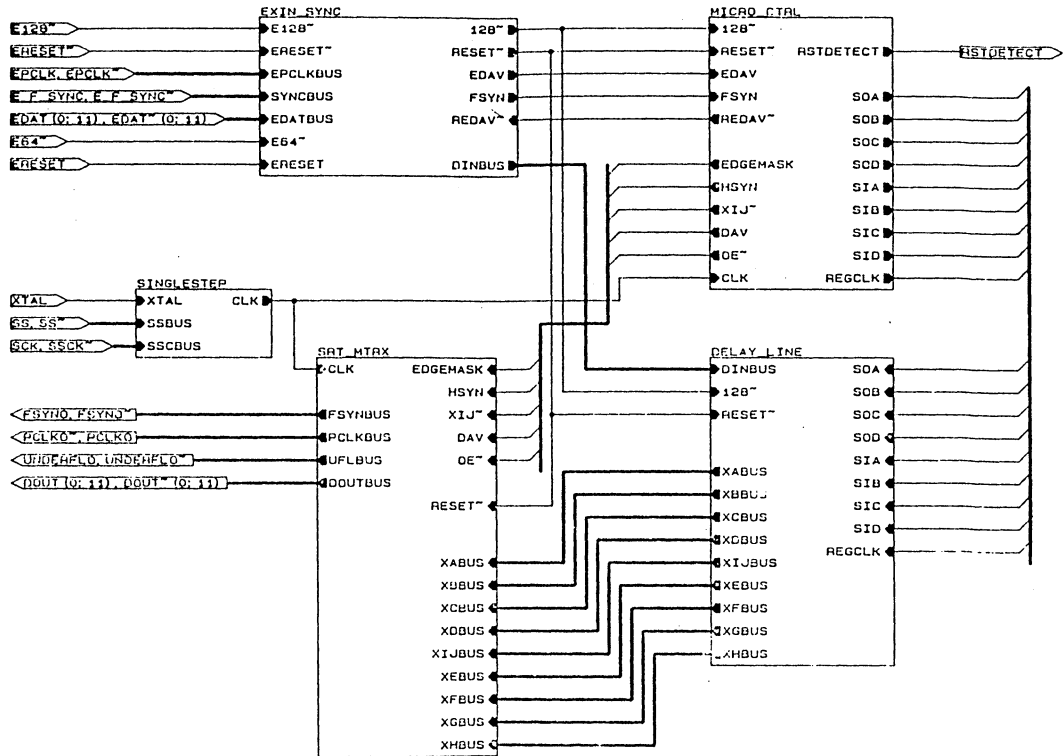
1. D.S. Fraedrich, "Analytic Evaluation of Frame Difference Processing on Terrain Clutter at MWIR Wavelengths," *SPIE* v. 781, *Infrared Image Processing and Enhancement* (1987), pp. 26-32.
2. D. Casasent, "Electro Optic Target Detection and Object Recognition," *SPIE* v. 762, *Electro-Optical Imaging Systems Integration* (1987), pp. 104-125.
3. B. Bhanu, A.S. Politopoulos, and B.A. Parvin, "Intelligent Autocueing of Tactical Targets," *SPIE* v. 435, *Architectures and Algorithms for Digital Image Processing* (1983), pp. 90-97.
4. B. Bhanu, "Evaluation of Automatic Target Recognition Algorithms," *SPIE* v. 435, *Architectures and Algorithms for Digital Image Processing* (1983), pp. 18-27.
5. M.S. Longmire, A.F. Milton, and E.H. Takken, "Simulation of Mid-Infrared Clutter Rejection. 1: One-Dimensional LMS Spatial Filter and Adaptive Threshold Algorithms," *Applied Optics*, v. 21, no. 21, Nov. 1982, pp. 3819-3833.
6. M.C. Hetzler, R.M. Smith, R.C. DuVarney, and J.M. Marks, "A Study of Clutter in Infrared Backgrounds," *SPIE* v. 781, *Infrared Image Processing Enhancement* (1987), pp. 10-17.
7. E.H. Takken, D. Friedman, A.F. Milton, and R. Nitzberg, "Least-Mean-Square Spatial Filter for IR Sensors," *Applied Optics*, v. 18, no. 24, Dec. 1979, pp. 4210-4222.
8. T.L. Bergen and P.K. Mazaika, "Evaluation of Spatial Filters for Background Suppression in Infrared Mosaic Sensor Systems," *SPIE* v. 341, *Real Time Signal Processing V* (1982), pp. 209-222.
9. M.A. Kruer, D.A. Scribner, and J.M. Killiany, "Infrared Focal Plan Array Technology Developments for Navy Applications," *Optical Engineering*, v. 26, no. 3, March 1987, pp. 182-190.

10. D.A. Scribner, M.R. Kruer, C.J. Gridley, and K. Sarkady, "Measurement, Characterization, and Modeling of Noise in Staring Infrared Focal Plane Arrays," *SPIE v. 782, Infrared Sensors and Sensor Fusion* (1987), pp. 147 - 160.
11. M.R. Pauli, M.S. Longmire, and E.H. Takken, "Point Source Detection with Multiple Spatial Filters in a Staring Array Sensor," *Proc. 33rd National Infrared Information Symposium*, May 1985.
12. T.F. Tao, D. Hilmers, B. Evenor, and D. Bar Yehoshua, "Focal Plane Processing Techniques for Background Clutter Suppression and Target Detection," *SPIE v. 178, Smart Sensors* (1979), pp. 2-12.
13. J.J. Otazo and R.R. Parenti, "Digital Filters for the Detection of Resolved and Unresolved Targets Embedded in Infrared (IR) Scenes," *SPIE v. 178, Smart Sensors* (1979), pp. 13-24.
14. C.M. Lo, "Forward Looking Infrared (FLIR) Image Enhancement for the Automatic Target Cues System," *SPIE v. 238, Image Processing for Missile Guidance* (1980), pp. 91-102.
15. R.T. Gray, D.G. McCaughey, and B.R. Hunt, "Median Masking Technique for the Enhancement of Digital Images," *SPIE v. 207, Applications of Digital Image Processing III* (1979), pp. 142-145.
16. J.W. Tukey, "Nonlinear (nonsuperposable) Methods for Smoothing Data," *Conference Record, 1974 IEEE EASCON*, p. 673.
17. J.W. Tukey, *Exploratory Data Analysis*, Addison-Wesely, Reading, MA, 1977 (preliminary edition 1971).
18. A.E. Beaton and J.W. Tukey, "The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data," *Technometrics*, v. 16, no. 2, May 1974, pp. 147-185.
19. G. Garibotto and L. Lambarelli, "Fast On-Line Implementation of Two-Dimensional Median Filtering," *Electronics Letters*, v. 15, no. 1, 4 January 1979, pp. 24-25.
20. L.R. Rabiner, M.R. Sambur, and C.E. Schmidt, "Applications of a Nonlinear Smoothing Algorithm to Speech Processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-23, Dec. 1975, pp. 552-557.
21. N.S. Jayant, "Average and Median-Based Smoothing Techniques for Improving Digital Speech Quality in the Presence of Transmission Errors," *IEEE Transactions on Communications*, v. COM-24, pp. 1043-1045, Sept. 1976.
22. T.S. Huang, G.J. Yang, and G.Y. Tang, "A Fast Two-Dimensional Median Filtering Algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-27, no. 1, Feb. 1979, pp. 13-18.
23. R. Steele and D.J. Goodman, "Detection and Selective Smoothing of Transmission Errors in Linear PCM," *Bell System Technical Journal*, v. 56, pp. 300-409, March 1977.

24. W.K. Pratt, "Median Filtering," in *Semiannual Report, USC IPI Report 620*, Image Processing Institute, University of Southern California, Los Angeles, CA, pp. 116-120, 1975.
25. W.K. Pratt, *Digital Image Processing*, Wiley-Interscience, New York, NY, 1978.
26. B.R. Frieden, "A New Restoring Algorithm for the Preferential Enhancement of Edge Gradients," *Journal of Optical Society of America*, v. 66, pp. 280-283, 1976.
27. M.I. Shamos, "Robust Picture Processing Operators and Their Implementation as Circuits," *Proc. ARPA Image Understanding Workshop*, Pittsburg, PA, pp. 127-129, Nov. 1978.
28. W.L. Eversole, D.J. Mayer, F.B. Frazee, and T.F. Cheek, Jr., "Investigation of VLSI Technologies for Image Processing," *Proc. ARPA Image Understanding Workshop*, Pittsburg, PA, pp. 191-195, Nov. 1978.
29. V.K. Aatre, E. Ataman, and K.M. Wong, "Median Filtering," *17th Allerton Conference*, Monticello, IL, Oct. 1979, pp. 886-895.
30. D.L. Knuth, *Sorting and Searching*, Addison-Wesely, Reading, MA, 1975.
31. V.E. Alekseyev, "Sorting Algorithms with Minimum Memory," *Kibernetika*, v. 5, pp. 99-103, 1969.
32. J.W. Tukey, "The Ninther, a Technique for Low-Effect Robust (Resistant) Location in Large Samples," in *Contributions to Survey Sampling and Applied Statistics*, ed. H.A. David, Academic Press, New York, NY, 1978, pp. 251-257.
33. D.R. Morgan, "Analog Sorting Network Ranks Inputs by Amplitude and Allows Selection," *Electron. Design*, v. 21, pp. 72-73, Jan. 1973. Also, "Correction," *Electron. Design*, v. 21, p. 1, Aug. 1973.
34. E. Ataman, V.K. Aatre, and K.M. Wong, "A Fast Method for Real-Time Median Filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-28, no. 4, Aug. 1980, pp. 415-420.
35. P.M. Narendra, "A Separable Median Filter for Image Noise Smoothing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. PAMI-3, no. 1, Jan. 1981, pp. 20-29.
36. D.J. Delman, "Digital Pipelined Hardware Median Filter Design for Real-Time Image Processing," *SPIE* v. 298, *Real-Time Signal Processing IV* (1981), pp. 184-188.
37. G.R. Arce and N.C. Gallagher, Jr., "State Description for the Root-Signal Set of Median Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-30, no. 6, Dec. 1982.
38. J.P. Fitch, E.J. Coyle, and N.C. Gallagher, Jr., "Median Filtering by Threshold Decomposition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-32, no. 6, Dec. 1984, pp. 1183-1188.

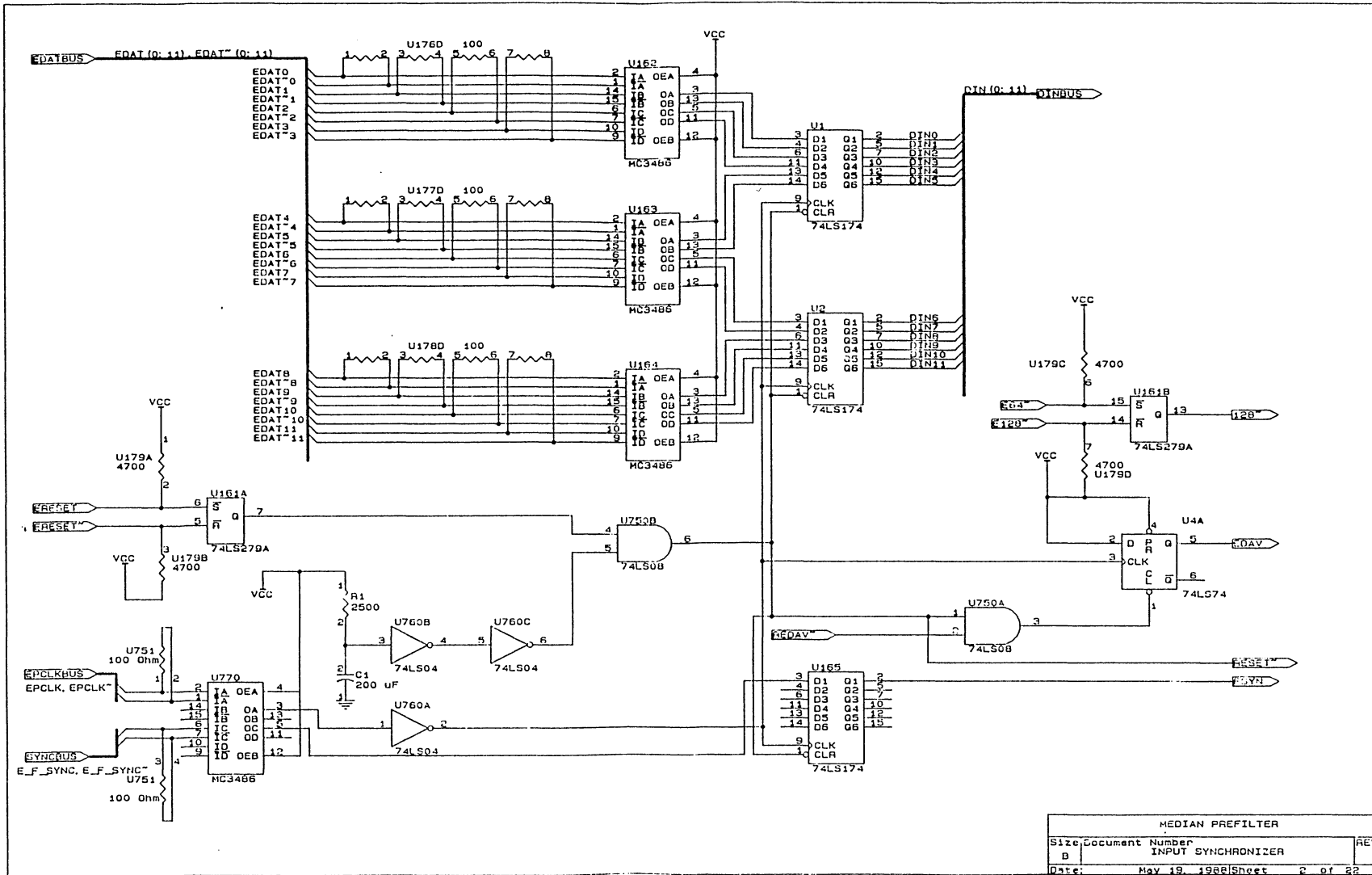
39. G.R. Arce, "Statistical Threshold Decomposition for Recursive and Nonrecursive Median Filters," *IEEE Transactions on Information Theory*, v. IT-32, no. 2, March 1986, pp. 243-253.
40. F. Kuhlmann and G.L. Wise, "On Second Moment Properties of Median Filtered Sequences of Independent Data," *IEEE Transactions on Communications*, v. COM-29, no. 9, Sept. 1981, pp. 1374-1379.
41. E. Ataman, V.K. Aatre, and K.M. Wong, "Some Statistical Properties of Median Filters," *IEEE Transactions on Acoustics Speech, and Signal Processing*, v. ASSP-29, no. 5, Oct. 1981, pp. 1073-1075.
42. N.C. Gallagher, Jr., and G.L. Wise, "A Theoretical Analysis of the Properties of Median Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-29, no. 6, Dec. 1981, pp. 1136-1141.
43. B.I. Justusson, "Median Filtering: Statistical Properties," in *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*, ed. T.S. Huang, Topics in Applied Physics v. 43, Springer-Verlag, New York, 1981, pp. 161-196.
44. P.F. Velleman, "Robust Non-Linear Data Smoothing," *Technical Report 89*, Ser. 2, Dept. of Statistics, Princeton University, 1977.
45. S.G. Tyan, "Median Filtering: Deterministic Properties," in *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*, ed. T.S. Huang, Topics in Applied Physics v. 43, Springer-Verlag, New York, 1981, pp. 197-217.
46. T.A. Nodes and N.C. Gallagher, Jr., "Median Filters: Some Modifications and Their Properties," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-30, no. 5, Oct. 1982, pp. 739-746.
47. A.C. Bovik, T.S. Huang, and D.C. Munson, Jr., "A Generalization of Median Filtering Using Linear Combinations of Order Statistics," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-31, no. 6, Dec. 1983, pp. 1342-1349.
48. T.A. Nodes and N.C. Gallagher, Jr., "The Output Distribution of Median Type Filters," *IEEE Transactions on Communications*, v. COM-32, no. 5, May 1984, pp. 532-541.
49. A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York, NY 1984.

# Appendix A. Schematics

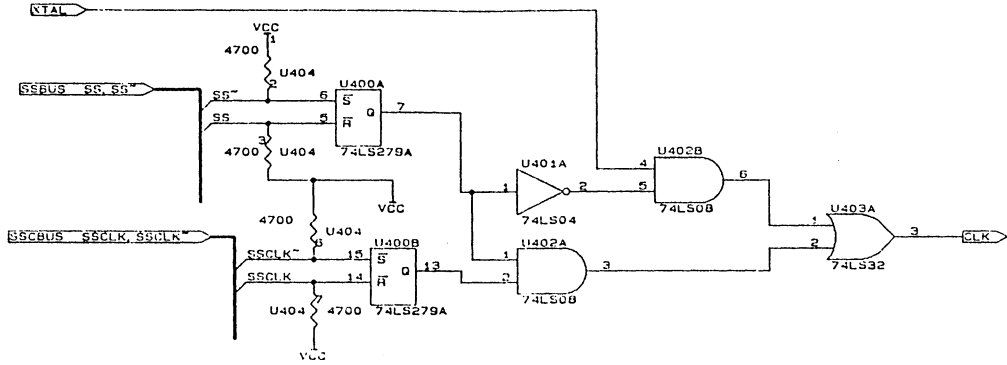


MEDIAH PREFILTER			
Size	Document Number	BLOCK DIAGRAM	REV
B			
Date:	May 19, 1998	Sheet	1 of 2

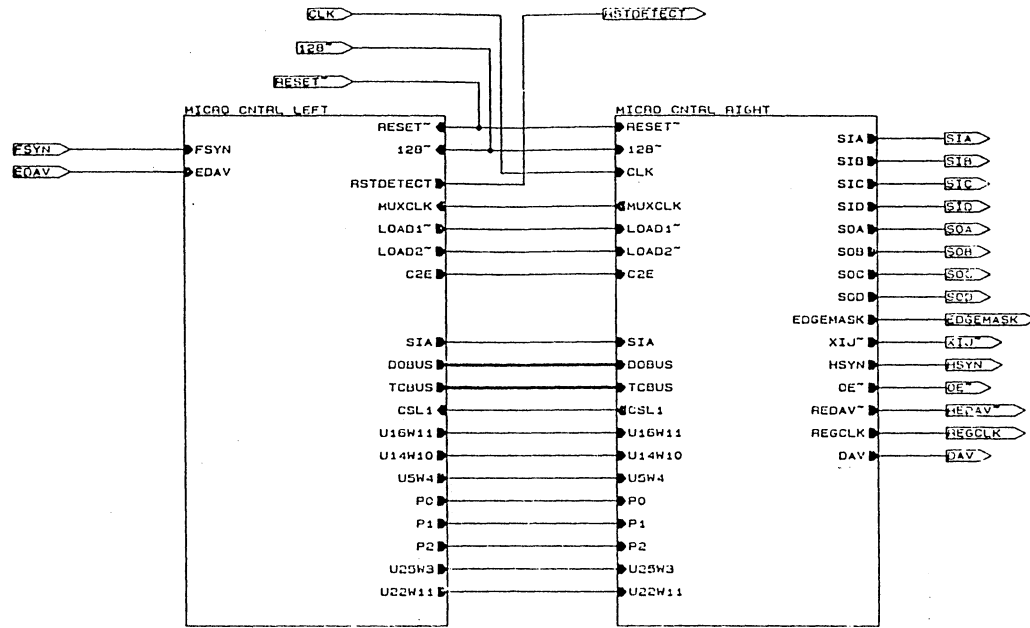




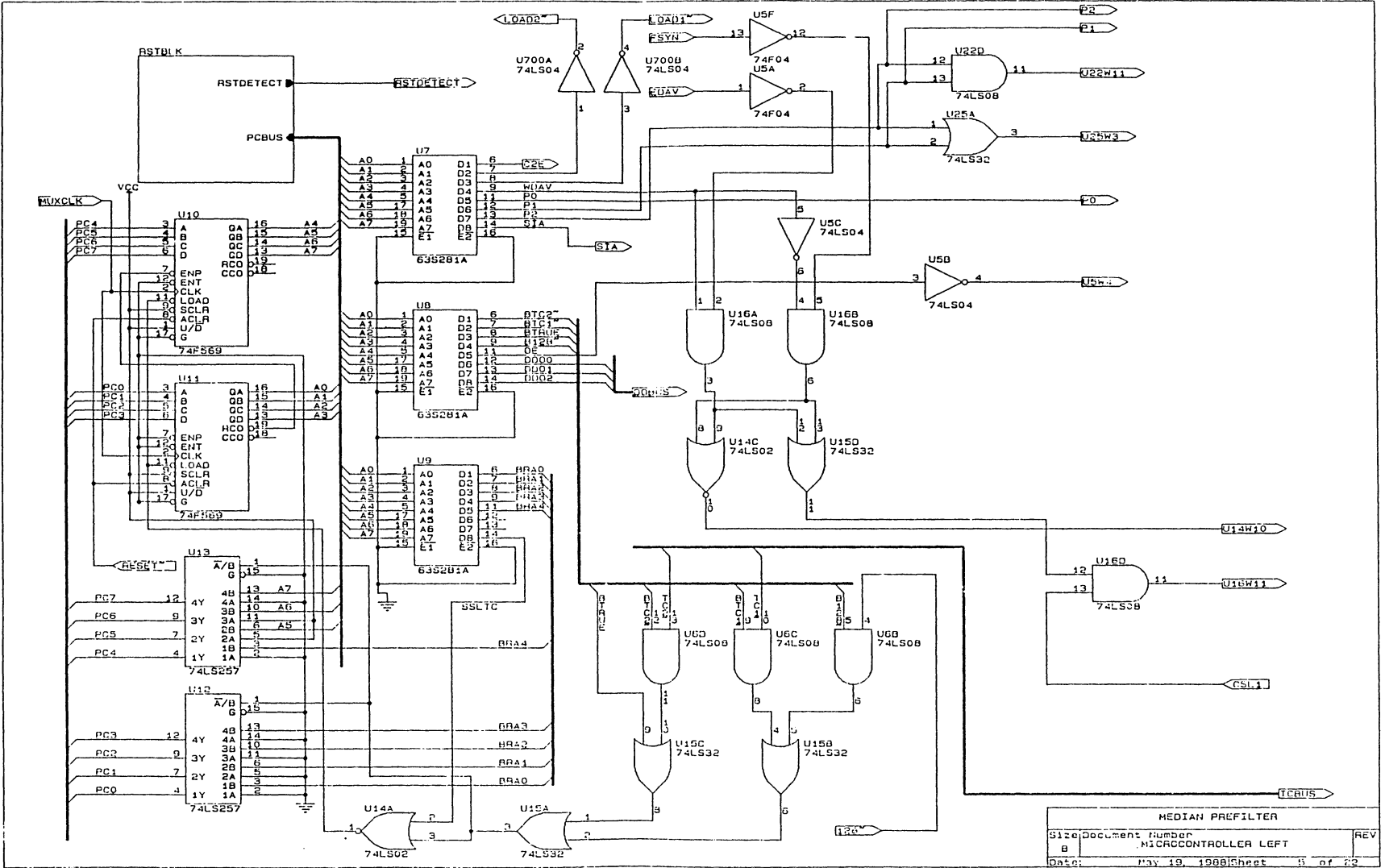
MEDIAN PREFILTER		
Size	Document Number	REV
B	INPUT SYNCHRONIZER	
Date:	May 19, 1998	Sheet 2 of 22



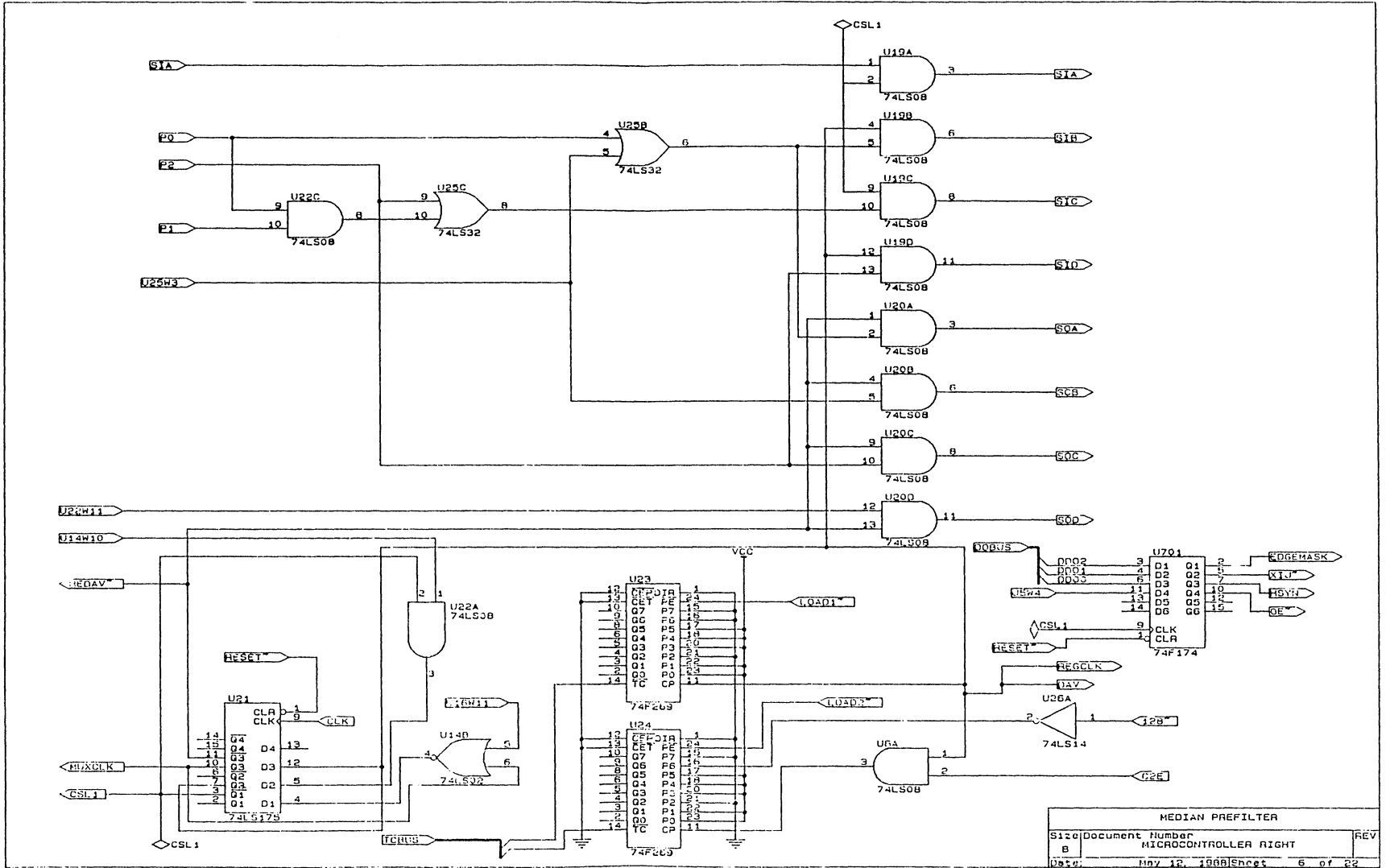
MEDIAN PREFILTER		
Size	Document Number	REV
B	SINGLESTEP	
Date:	May 13, 1998	Sheet 3 of 22



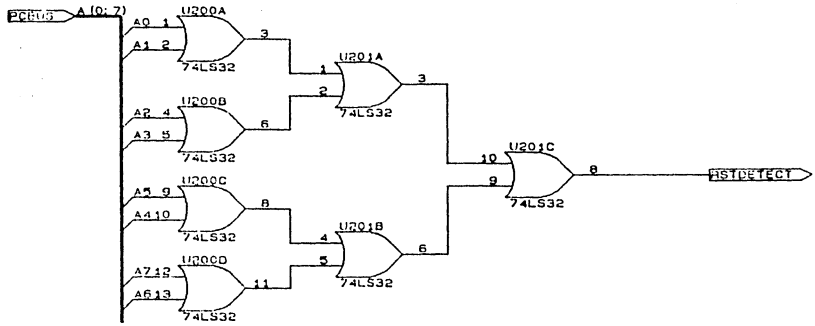
MEDIAN PREFILTER		
Size	Document Number	REV
B	MICROCONTROLLER BLOCK DIAGRAM	
Date:	May 19, 1988	Sheet 4 of 22



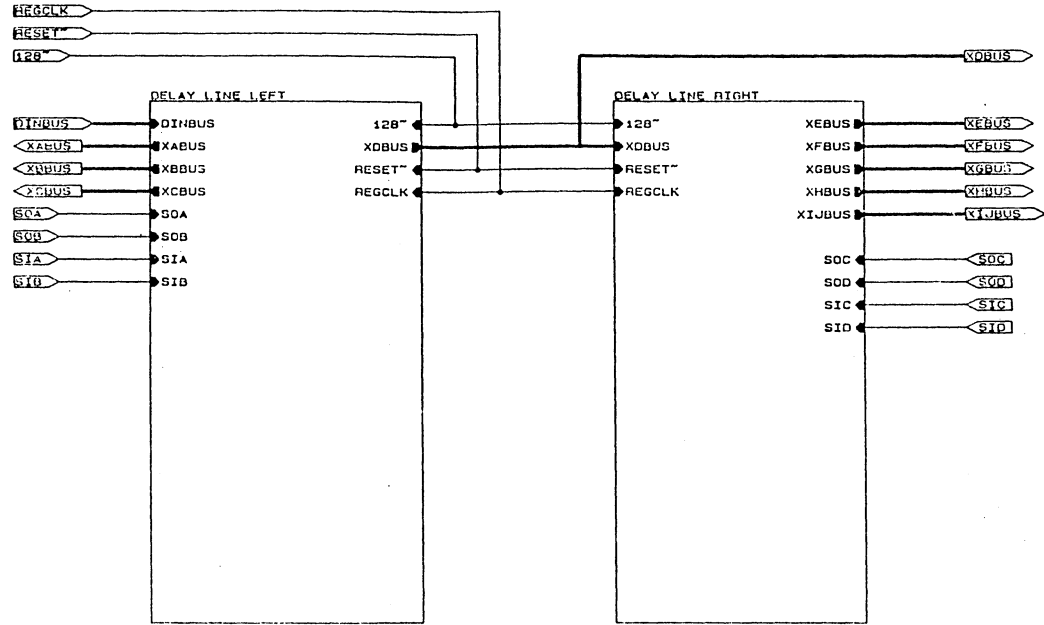
MEDIAN PREFILTER			
Size	Document Number	REV	
B	MICROCONTROLLER LEFT		
Date:	May 19, 1988	Sheet	5 of 22



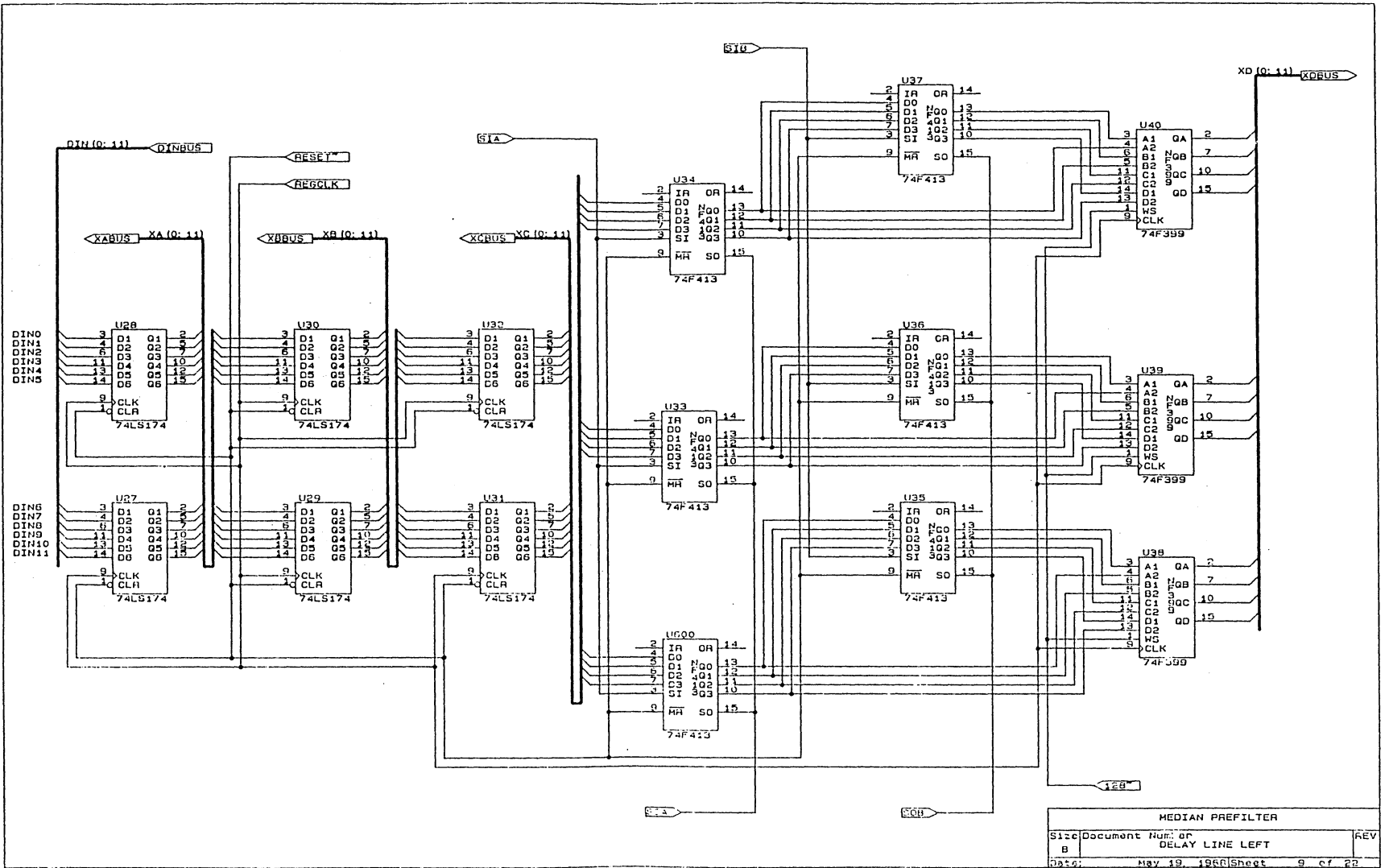
MEDIAN FILTER			
Size	Document Number		REV
B	MICROCONTROLLER RIGHT		
Date	May 12, 1988	Sheet	6 of 22



MEDIAN PREFILTER		
Size:	Document Number	HEV
0	RESET DETECT	
Date:	May 19, 1988	Sheet 7 of 32

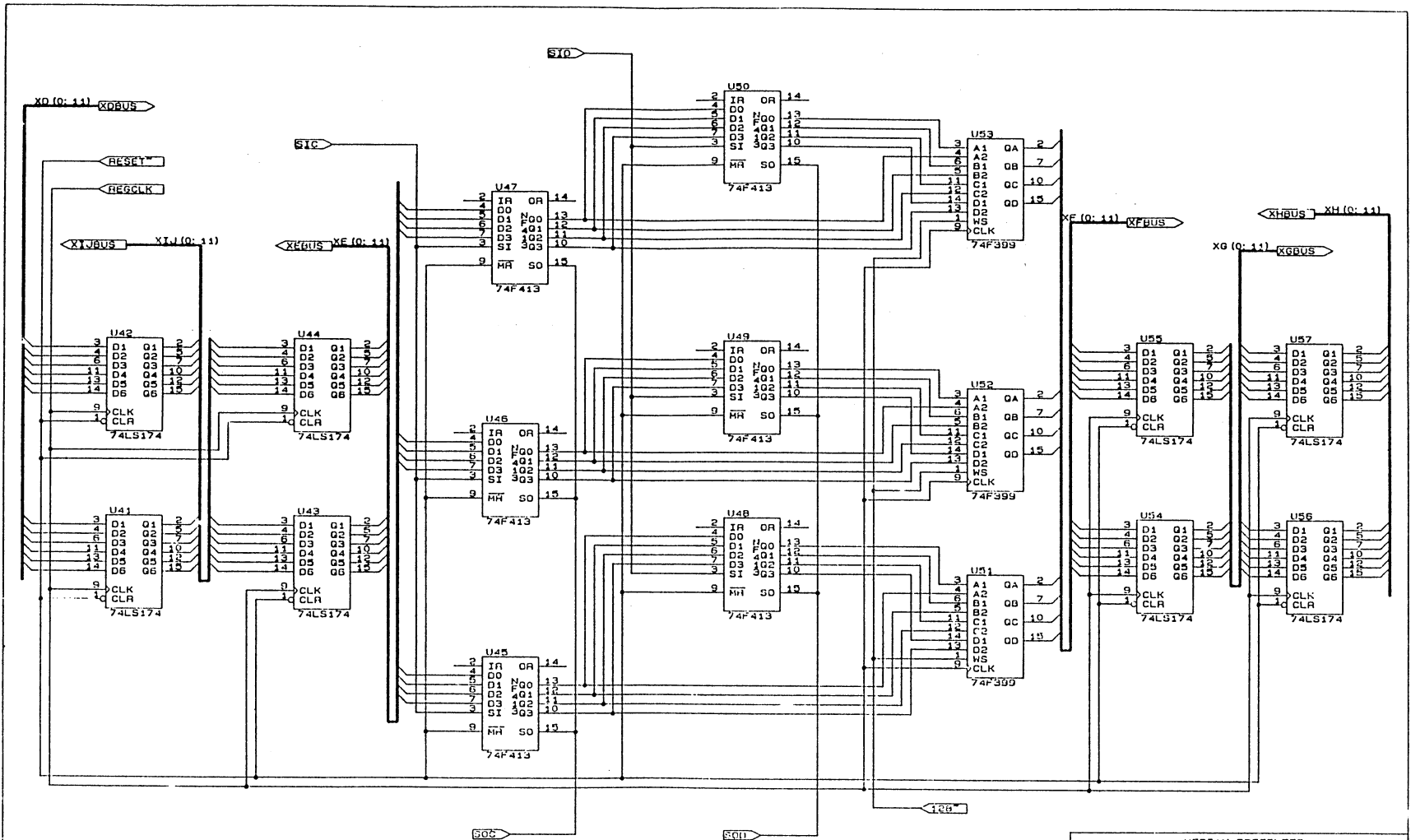


MEDIAN PREFILTER		
Size	Document Number	REV
0	DELAY LINE BLOCK DIAGRAM	
Date:	May 19, 1988	Sheet 0 of 23

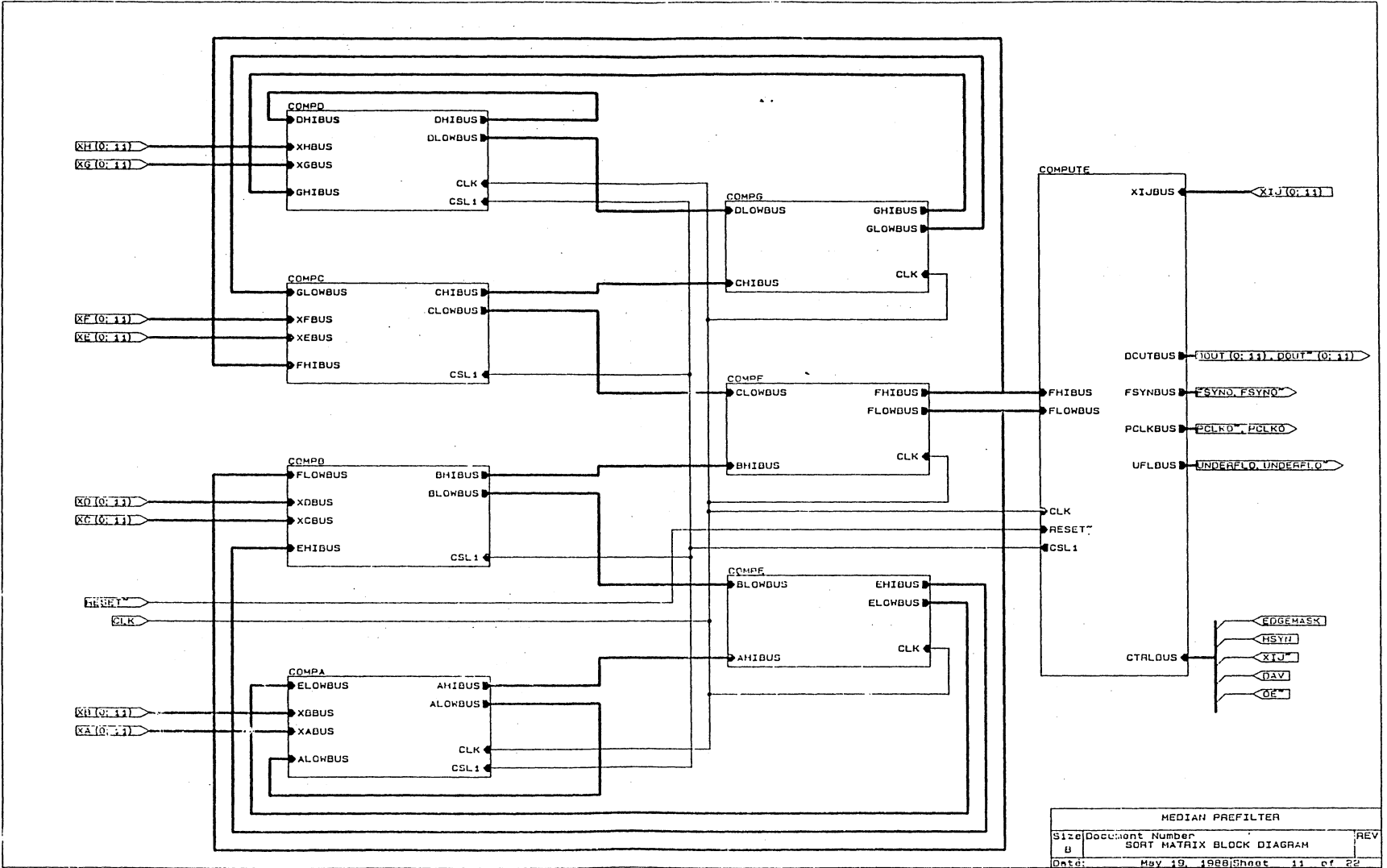


MEDIAN PREFILTER			
Size	Document	Num of	REV
B		DELAY LINE LEFT	
Date:	May 19, 1966	Sheet	9 of 22

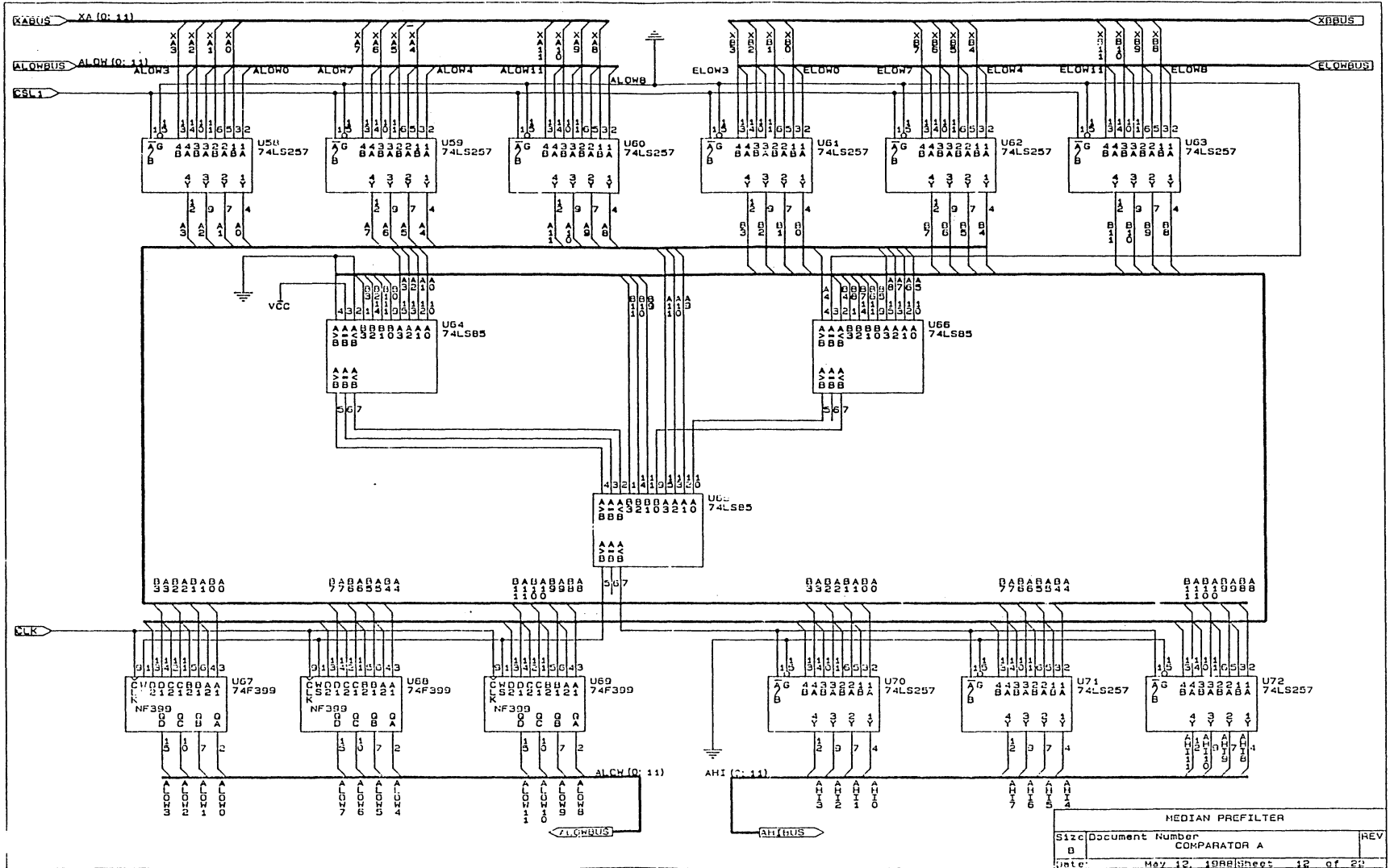




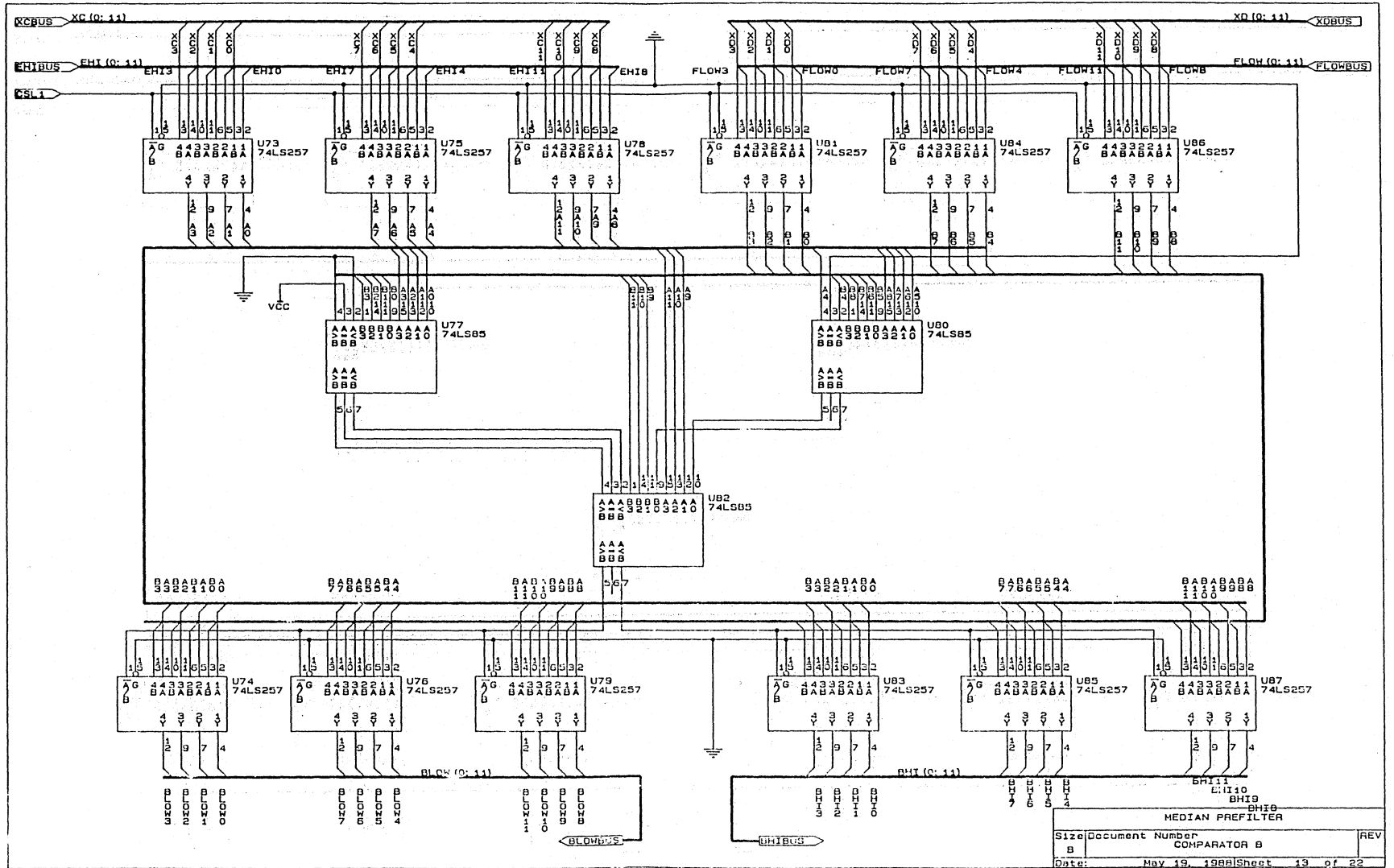
MEDIAN PREFILTER			
Size	Document Number		REV
B	DELAY LINE RIGHT		
Date:	May 19, 1988	Sheet	10 of 22



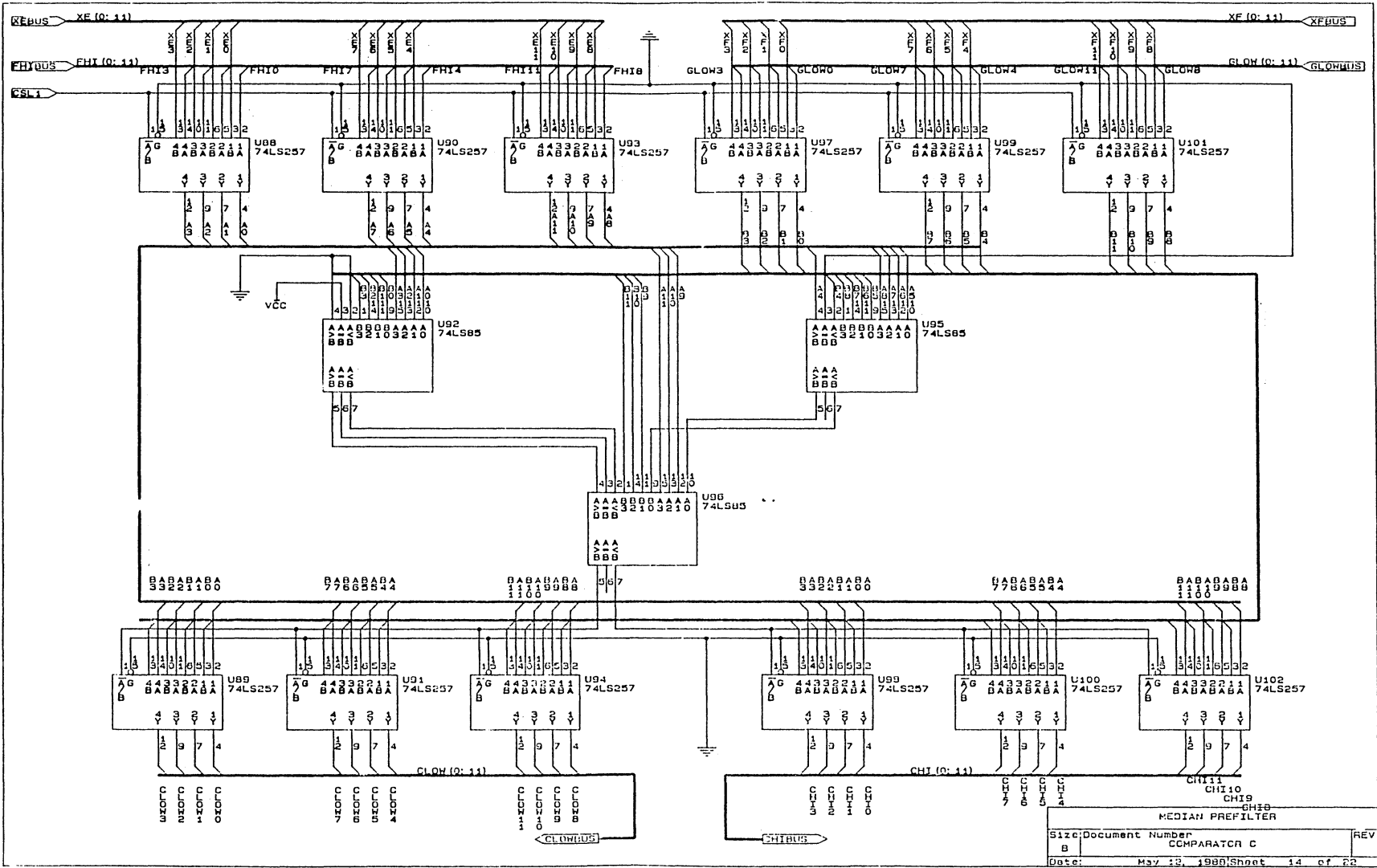
MEDIAN PREFILTER		
Size	Document Number	REV
B	SORT MATRIX BLOCK DIAGRAM	
Date:	May 19, 1988	Sheet 11 of 22



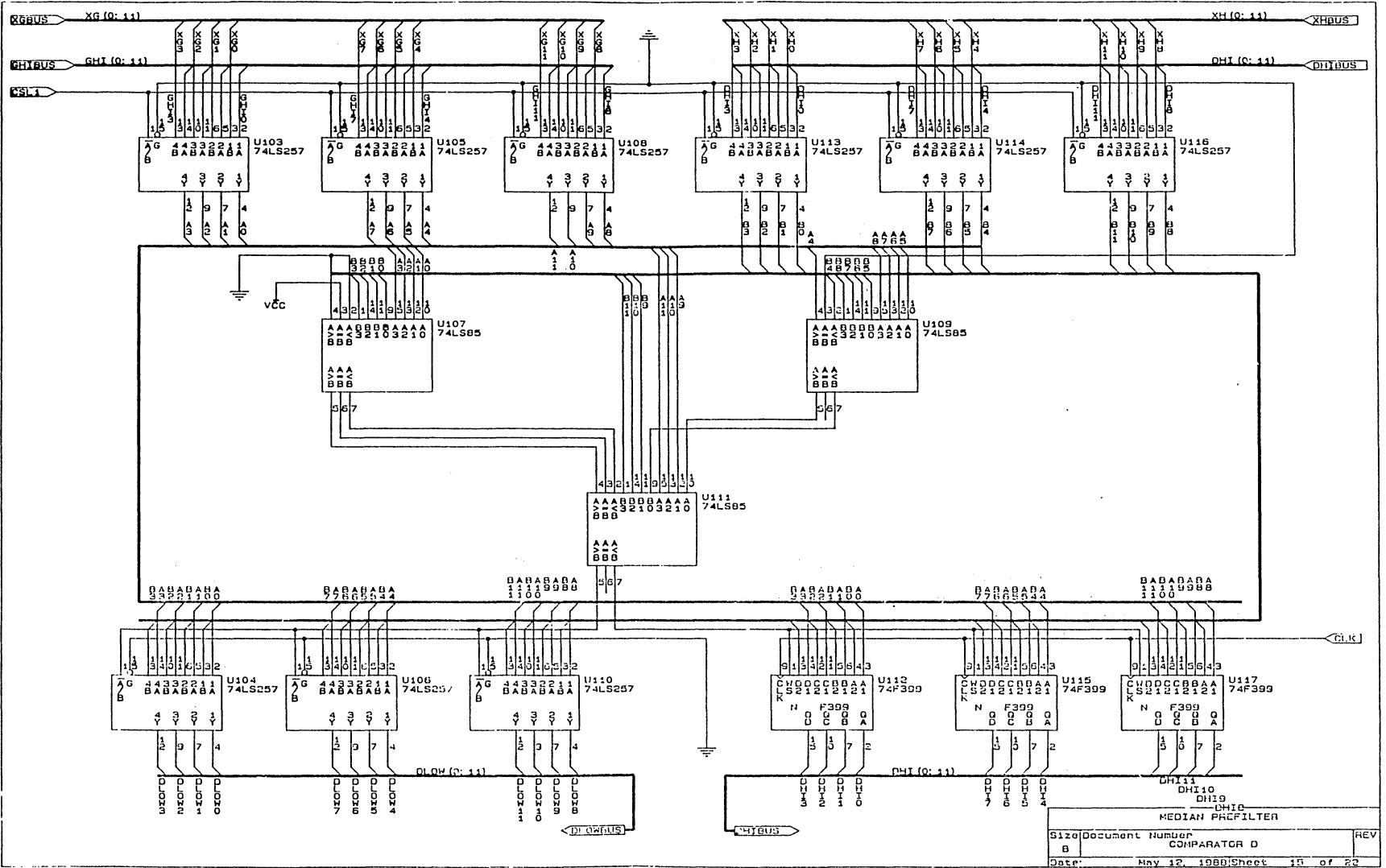
MEDIAN PREFILTER		
Size	Document Number	REV
B	COMPARATOR A	
Date	May 12, 1988	Sheet 12 of 21

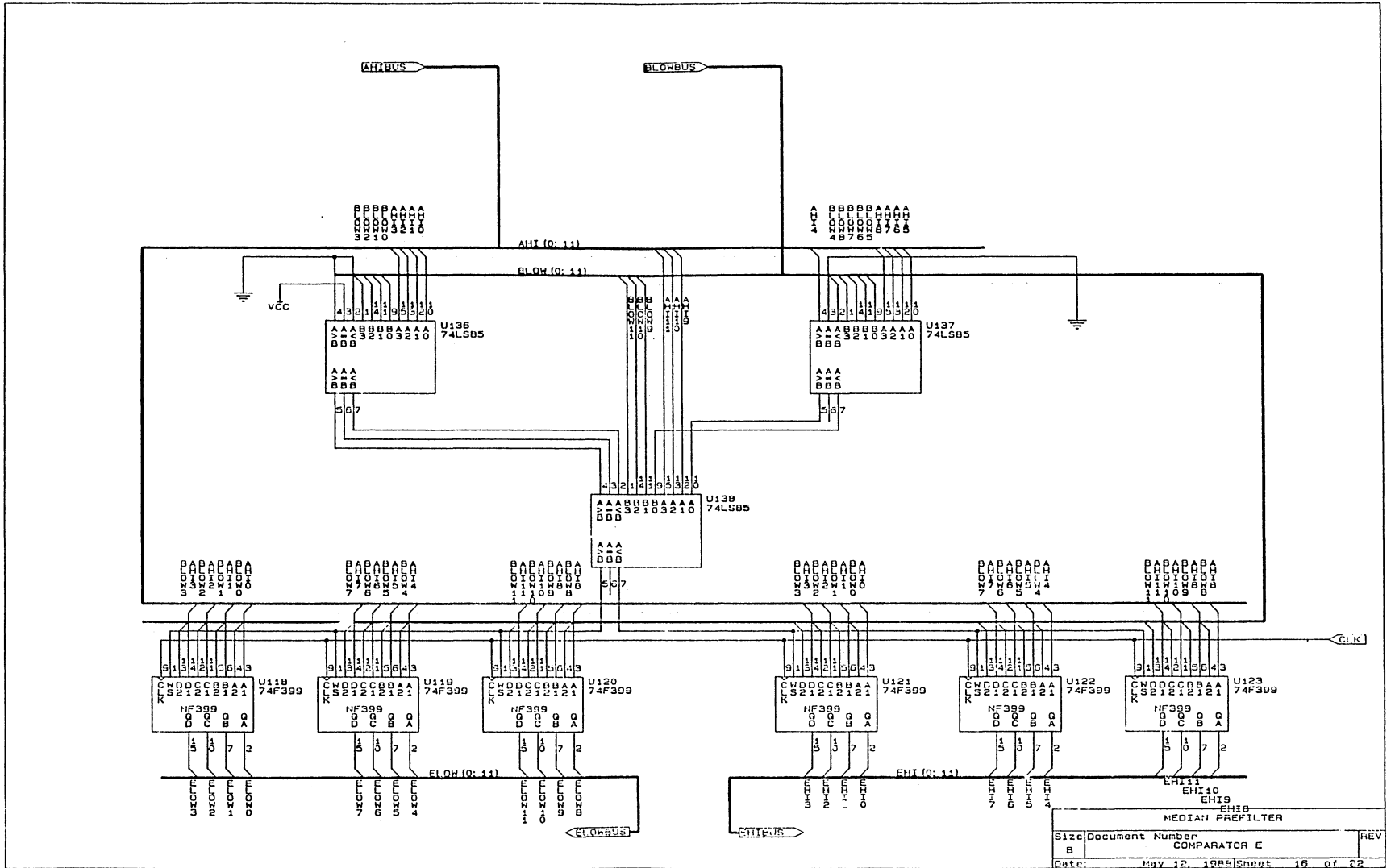


Size	Document Number	REV
B	COMPARATOR B	
Date:	May 19, 1988	Sheet 19 of 22

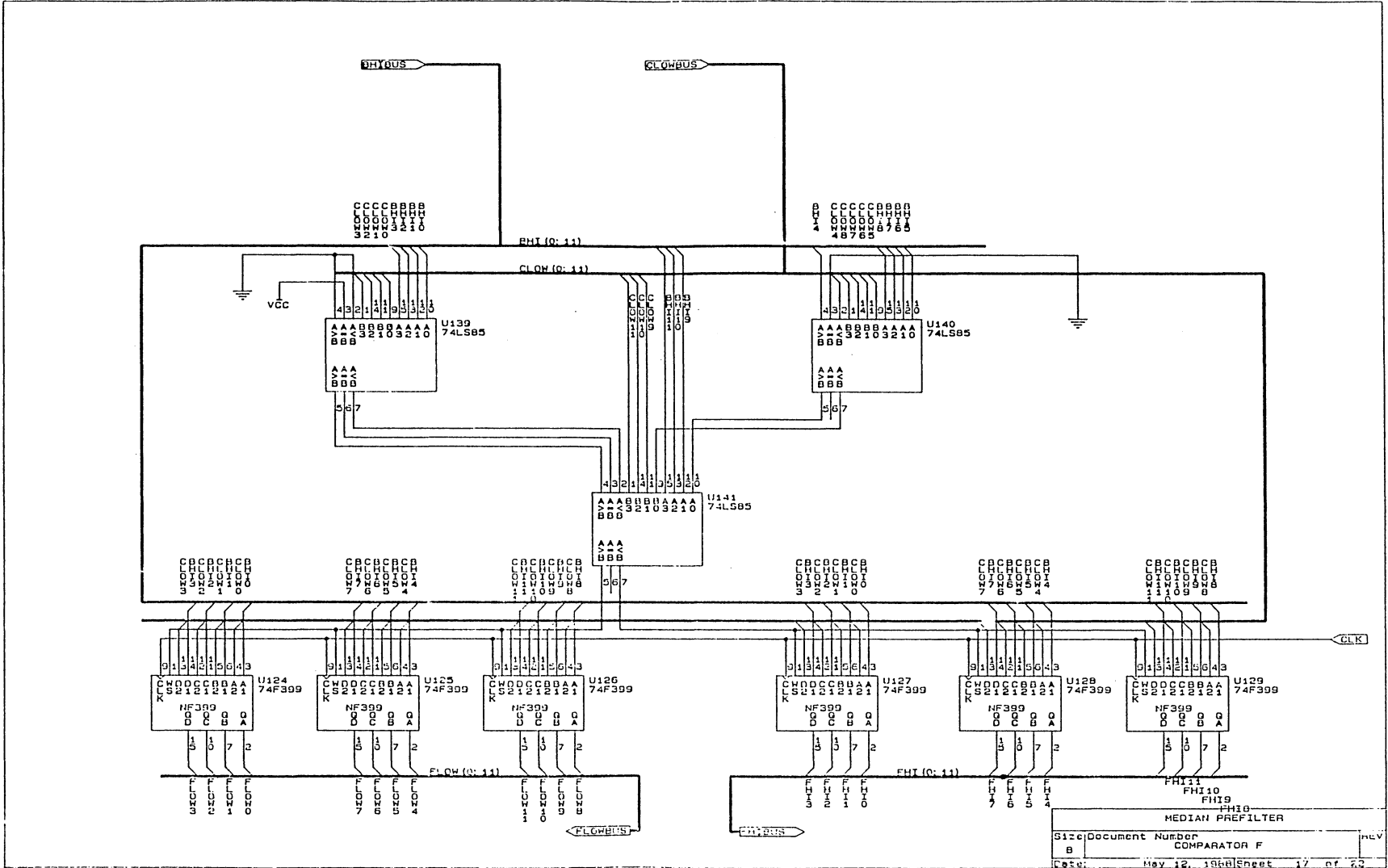


Size: Document Number		REV
B		COMPARATOR C
Date:	May 13, 1988	Sheet 14 of 22

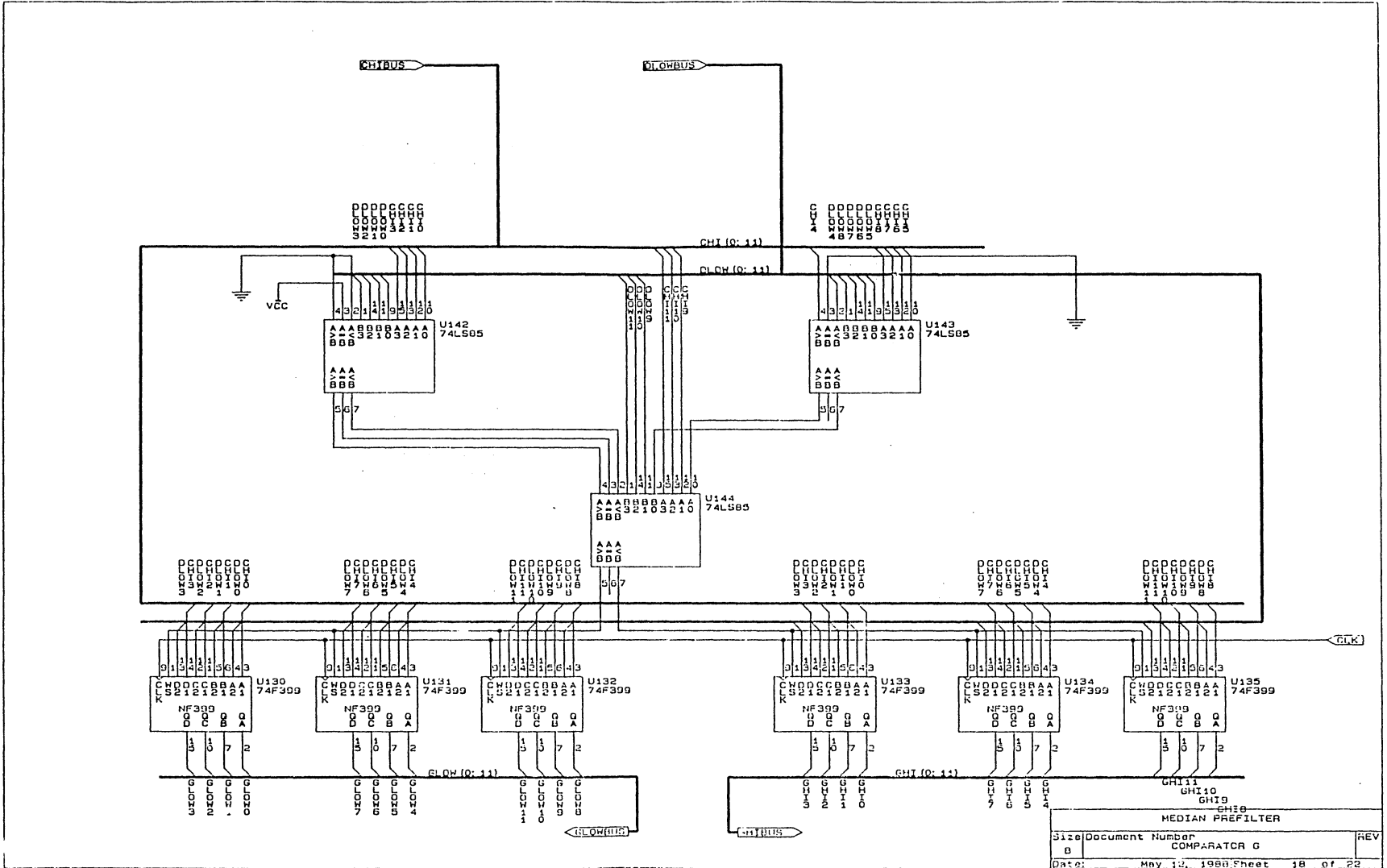




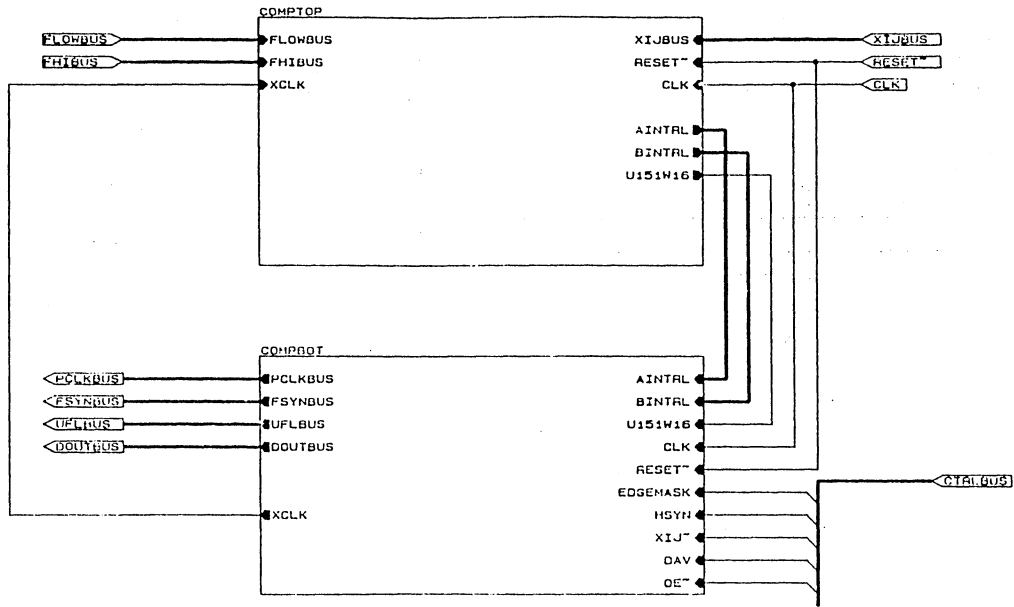
Size Document Number		REV
B		COMPARATOR E
Date:	May 12, 1995	Sheet 16 of 22



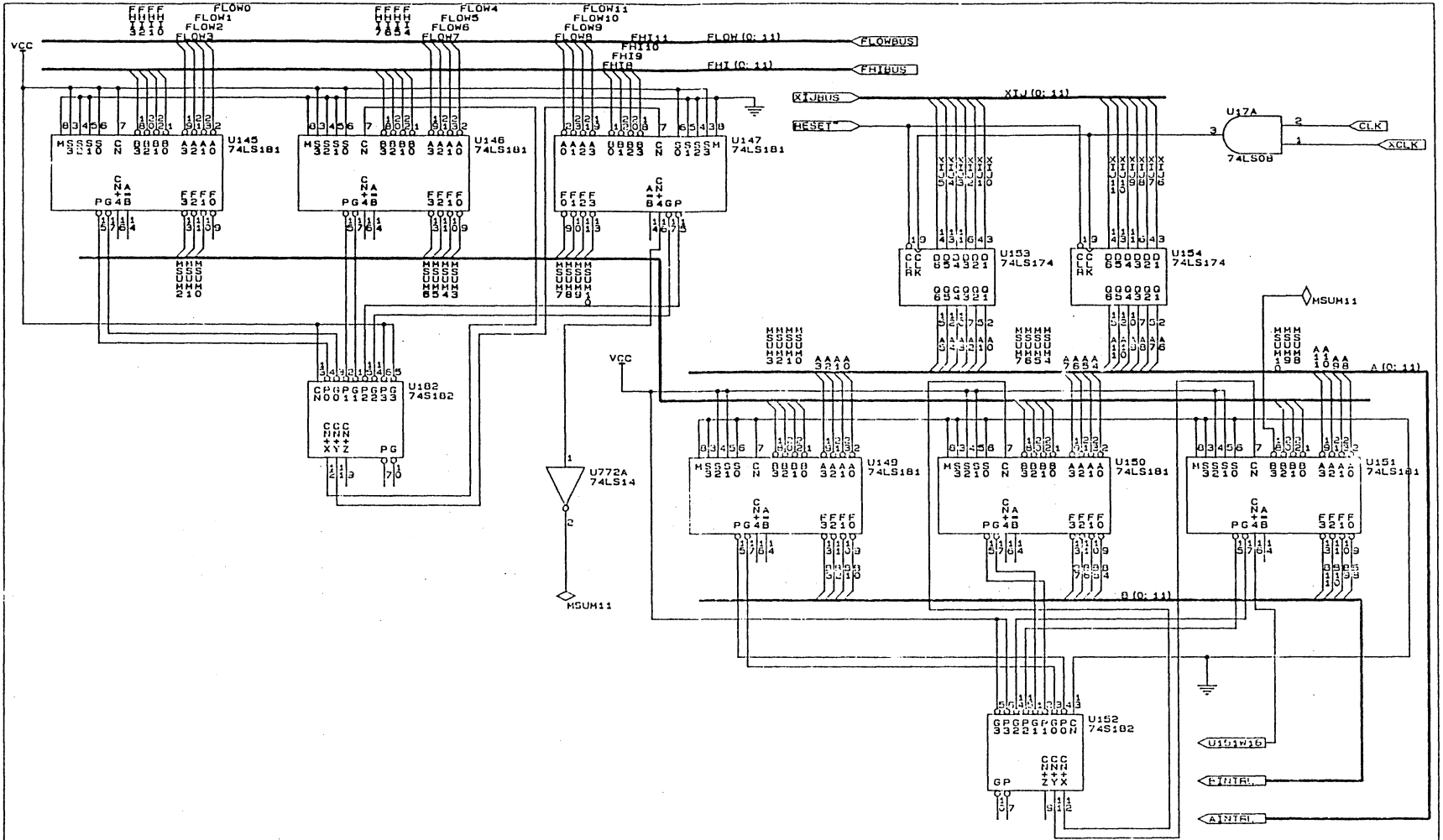




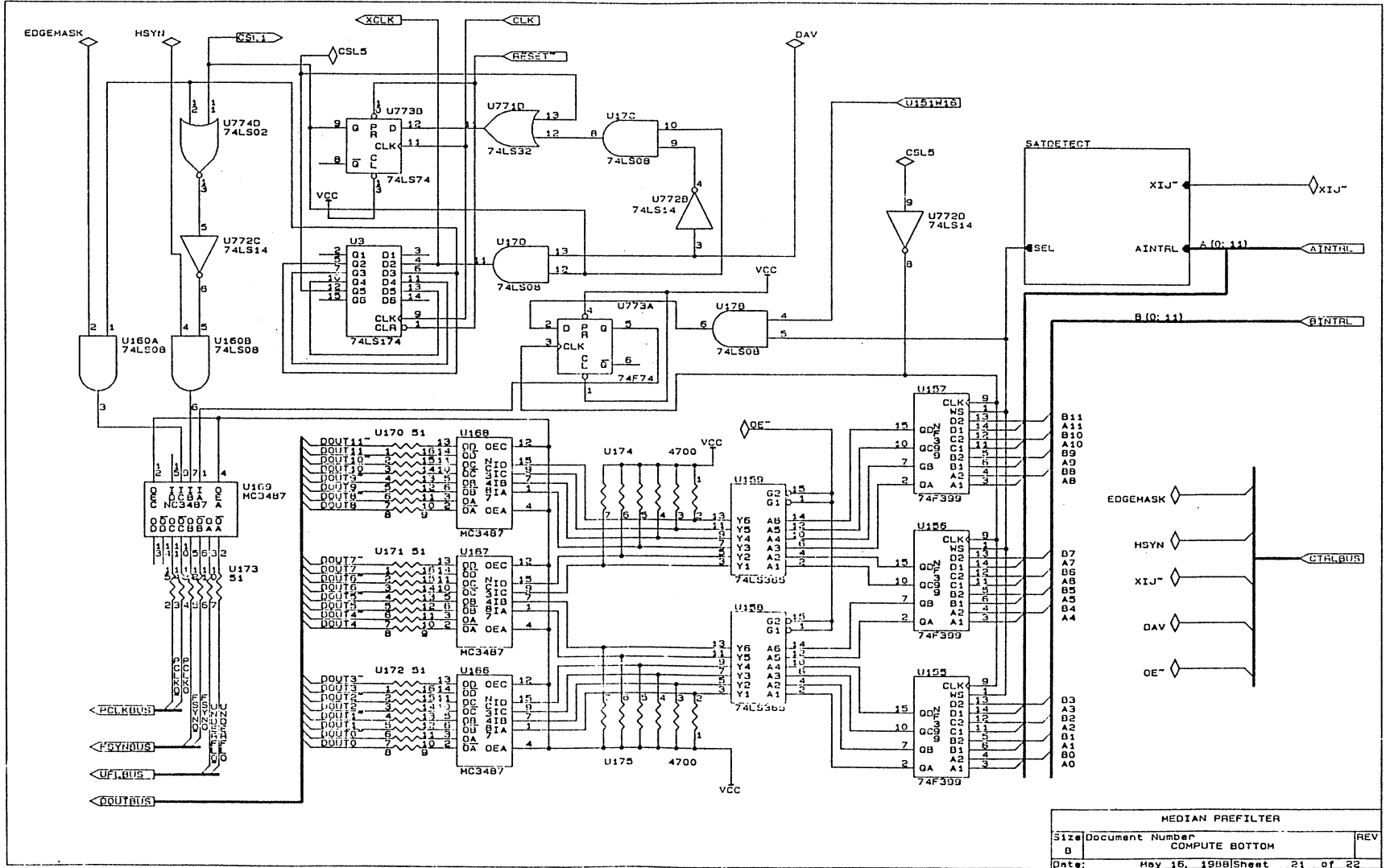
Size	Document Number	REV
B	COMPARATOR G	
Date:	May 14, 1988	Sheet 18 of 22

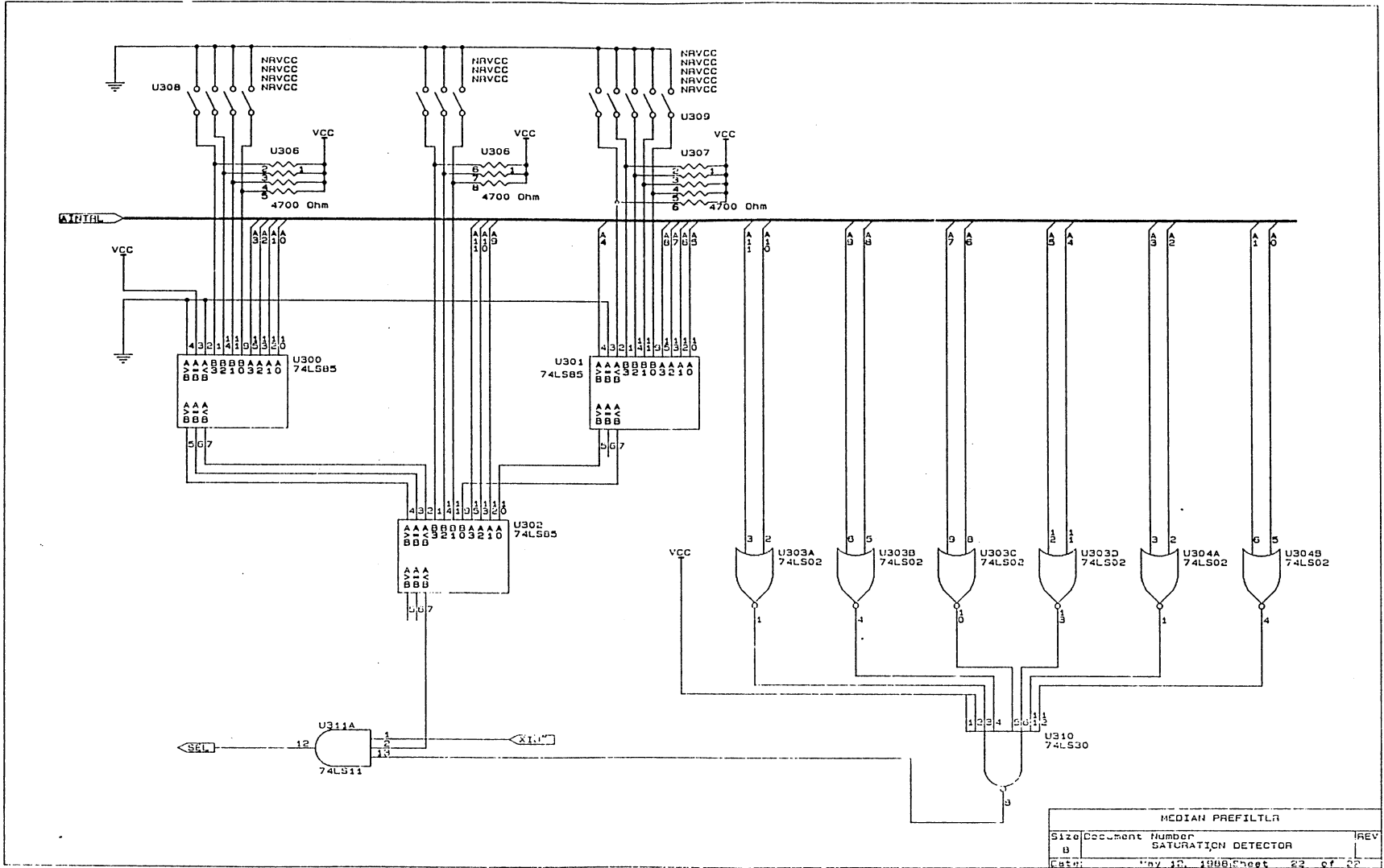


MEDIAN PREFILTER		
Size	Document Number	REV
8	COMPUTE MODULE BLOCK DIAGRAM	
Date:	May 19, 1988	Sheet 19 of 22



MEDIAN PREFILTER		
Size	Document Number	REV
B	COMPUTE TOP	
Date:	May 12, 1988 Sheet 20 of 22	





MEDIAN PREFILTER			REV
Size	Document Number		
U	SATURATION DETECTOR		
File:	May 10, 1988	Sheet 22 of 22	

**The vita has been removed from  
the scanned document**