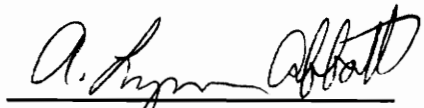
Real-time Parameter Adjustment for Archival Image Scanning

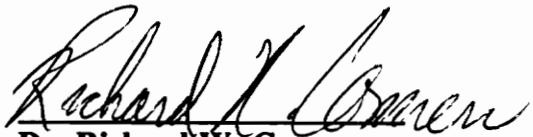
By Brian S. Cruikshank

To fulfill the thesis requirement for
Master of Science in Electrical Engineering Degree
Virginia Polytechnic Institute and State University
Bradley Department of Electrical Engineering

Approved:

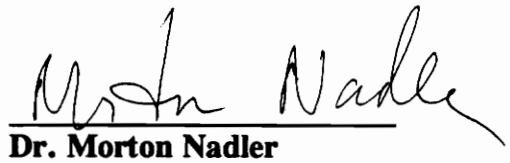


Dr. A. Lynn Abbott



Dr. Richard W. Connors

Dr. Scott F. Midkiff



Dr. Morton Nadler

June 3, 1992

Abstract

Many older documents are of poor quality and are deteriorating with the passage of time. Furthermore, many of these documents are currently stored in the form of paper or photographic film, and therefore storage space, accessability, and security are serious problems. Because of these problems, many older documents are being restored and converted to digital form for archival. Image scanners which perform these tasks must be fast and must produce digital images that are of high quality.

A Scan Optimizer was developed by Image Processing Technologies, Inc., to assist in the restoration and conversion processes. This device dramatically improves the throughput of image scanners by the use of high-speed complicated image processing. As originally designed, however, the IPT Scan Optimizer requires the manual adjustment of several parameters to obtain the best results for a given document. Because of this manual adjustment, human intervention was often required, and conversion speeds were drastically reduced. In order to remove the need for manual adjustments, an Automatic Parameter Setting (APS) algorithm was developed. This algorithm relies on histogram analysis over nonoverlapping image regions for the computation of scan parameters.

This thesis describes a hardware realization of the APS algorithm. To achieve a real-time implementation that is low in cost, several compromises were necessary in the design. This two-board set is compatible with the AT bus, and is composed of one SPARC RISC chip, four Xilinx Programmable Logic Cell Arrays, ten PALs and many RAMs and supporting logic. This system has been designed, built, and successfully tested with many documents.

Acknowledgements:

I would like to thank a number of people that made this possible.

- First of all, thanks to Jim Tatem and Dr. Morton Nadler for the invention of the Automatic Parameter Setting algorithm which is the basis of this thesis.

- Thanks to all the people at Image Processing Technologies for any assistance they have given to me during this project.

- Thanks to Image Processing Technologies for funding the project from concept to final hardware.

- Thanks to all my friends who kept me from going crazy.

Table of Contents

List of Figures

1. Introduction

1.1. Document Conversion

Yesterday's society was one of paper and film; tomorrow's will be computerized; today's involves change, restoring and converting documents. The conversion to digital storage media is not a simple task due to the large number of archives which exist and continue to grow. Billions of important documents are stored at many different places such as banks, county record offices, hospitals, and insurance companies. Most of these documents are legal or historic in nature, and this requires them to be accessible and of high quality. Using yesterday's archiving methods, documents are often poor in quality and sometimes degrade over time. New conversion methods and computerization will not only improve this situation, but will also greatly conserve space and improve accessibility [Stucki 1978].

In many cases, yesterday's archives are poor representations of the originals. Most archiving was done by film, often either microfilm or microfiche. If the exposure was not set properly during the filming process, the archived document is virtually unreadable. Most archiving film was also high in contrast, which made the exposure more critical and the output more problematic if the exposure was set wrong. So that they can be usable again, many of these film records need to be restored using special image processing techniques.

Some of the older archives are also degrading, and therefore need to be restored into a different format while it is still possible. A good example of this is photostats. The photostat process is a photographic process; it involves special paper and chemicals. The

paper must undergo a process similar to developing. If the process is not performed correctly, the photostats fade as old photographs often do. When these photostats are restored, it is very important to use the most accurate methods possible, because soon they may be unrestorable.

Archive storage space is another large problem which will be alleviated in part by conversion to digital form. Instead of finding better archiving methods, many facilities are merely attempting to pack more documents into the same room. This does not address the root of the problem and makes access to the documents much more difficult. The conversion to digital form offers the potential for tremendous improvement in space utilization and accessibility.

The conversion process starts by converting the documents to digital images, which are then compressed and stored. With today's technology, the contents of an entire filing cabinet (20,000 images) can fit onto one 5 1/4" optical disk. On many occasions these images are converted to ASCII form using optical character recognition (OCR). In these cases, the images must be very accurate and virtually free of noise so that recognition rates are high. This puts great emphasis on the image conversion process, but the potential rewards are great: 1,000,000 to 2,000,000 pages on one disk. Due to this incredible saving of space, multiple copies of the archives can be made, and this redundancy makes the information safer. Also, using a jukebox disk storage and retrieval system, archivists can have easy access to billions of documents in the space of one or two file cabinets.

1.2. The Conversion System

A conversion or restoration system in most cases consists of a host computer, a scanner, a printer, a mass storage device, and other related hardware and software. Figure 1.1 shows the relationship of the components. The scanner is the input device where images on paper or film are digitized. The output is usually a binary image which is then sent to the computer where it is often compressed into a CCITT Group IV format [National Archives 1991]. This image may then be displayed on a monitor, reprinted, or stored on a mass storage device for later retrieval.

Because of the numbers of documents needing to be converted, conversion systems are usually designed to operate at high speeds. A scanner can typically process an 11" x 17" page in approximately four seconds. Hardware boards are now available to do the Group IV compression at similar rates. The printer and mass storage devices operate at speeds similar to the scanner and can operate in parallel with scanning. Theoretically, therefore, hardware conversion can take place at approximately fifteen pages per minute. However, this usually is not possible in practice due to the need for human inspection and rescans. Scanners have variable parameters to control their outputs and must be adjusted for different document types and qualities. Because of this, rescans are often necessary. Each page must be visually checked, and if there is a problem, the scanner parameters must be adjusted and a rescan performed. This has reduced the typical conversion speed to three pages per minute. With only a million documents, the difference is already 230 days with human intervention versus 40 days without it. Obviously, for a billion documents you must totally eliminate inspection.

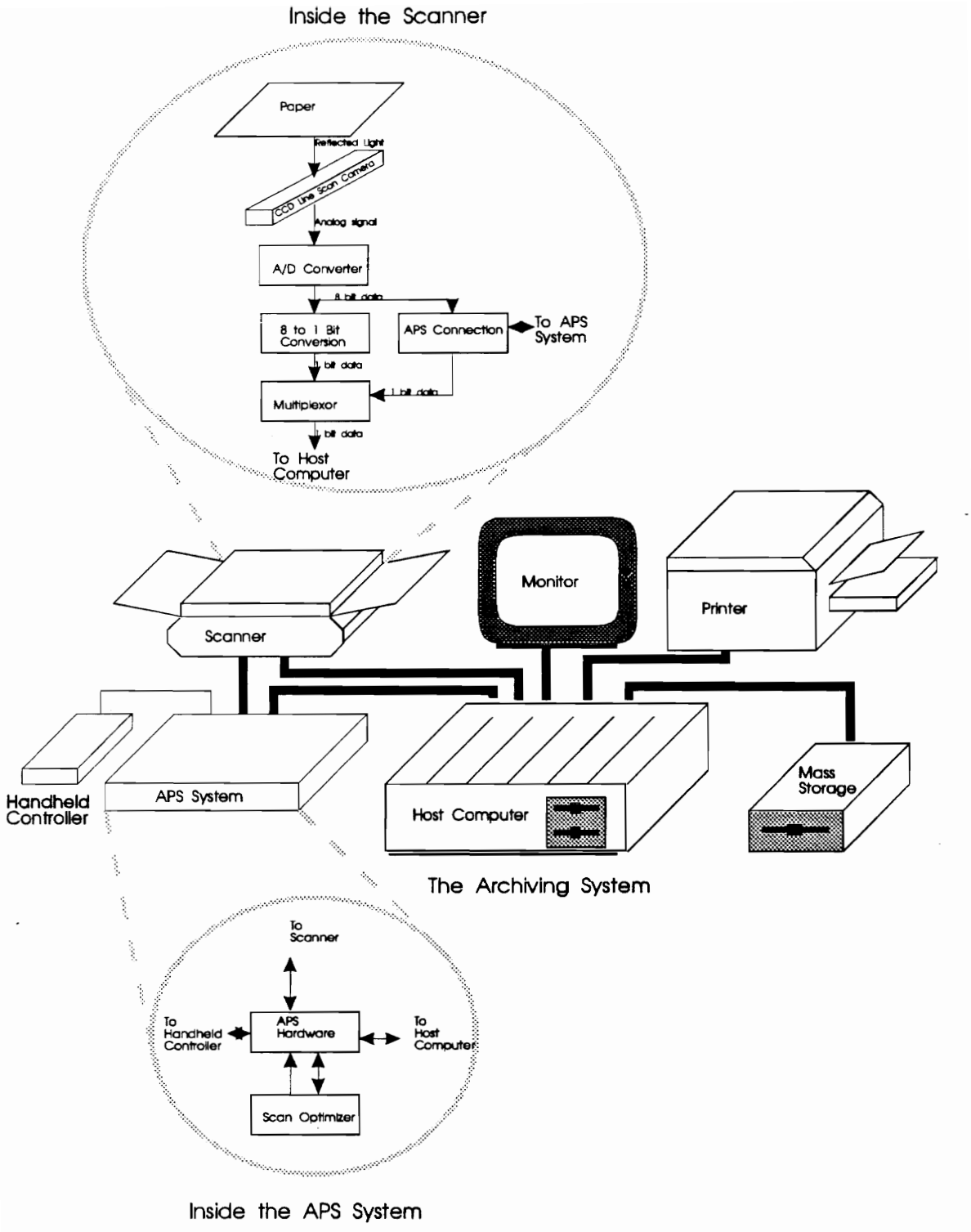


Figure 1.1. Inside the archiving system.

1.3. The Document Scanner

The scanner is the vital component that is being considered in this thesis. To make large scale archiving reasonable, throughput must be increased by removing or reducing the need for human intervention. The only way this can happen is by removing, or at least reducing, the need for manual parameter adjustment.

There are several electrical components inside the scanner; the interconnection of these components is described in Figure 1.1. The line CCD sensor detects the amount of light being given off by a line of the document and converts this to an analog value per pixel along the line. This line sensor is physically moved down the page to scan the entire page. As the sensor moves, its output signal is transformed into the digital domain by an analog-to-digital (A/D) converter. This converter usually outputs either 6 or 8 bits of data per sample. Since this is an enormous amount of data for one document even when compressed, and because most documents are strictly black and white, the data are usually translated into one bit per sample. This conversion to binary form is the critical process where the scanner parameters operate.

1.4. 8-bit to Binary Conversion

Accurately portraying the original document in a binary image is a difficult task. Characters and symbols are often very nearly being connected or broken in the original and these subtleties are destroyed in the conversion process. These problems are more serious than one would think since our eyes recognize characters and symbols and automatically break or connect the pieces. Also a document which has faded or which has

been poorly reproduced will suffer from variations in background intensity and from poor contrast.

In the past, 8-bit to binary conversion has been a simple threshold operation or variation of that. These are very crude processes which do not work well with poor documents and require much parameter adjustment. Other methods of image processing algorithms have been developed, but most of these include Fourier transforms which take enormous amounts of time even using high speed processors. The realization of these algorithms into near real-time hardware is impossible for a reasonable price.

A few simpler algorithms have been implemented in hardware. In some cases, 8-bit preprocessing filters have been developed to improve the conversion process. Background compensation [Ricoh 1986], texture removal [Ricoh 1986B], smoothing [Wang et al. 1981], and modulation transfer function (MTF) correction [Ricoh 1986] are all 8-bit preprocessing algorithms that have been implemented in hardware. Background compensation is usually an analog process that removes background noise, but does not improve any faint information levels. Sometimes, if the information level is at a similar contrast level as the background, the information can inadvertently be removed. Texture removal is similar to background compensation, but removes the background that the CCD and lights artificially impose. MTF stretches the dynamic range in the 8-bit domain; however, while it increases detail, it also increases noise. All of these algorithms can help improve the image, but they still leave the 8-bit to binary conversion to another algorithm.

Dynamic thresholding [Fujitsu 1991] and Laplacian filtering with a threshold [Gonzalez et al. 1987, Offen 1985] are two 8-bit to binary algorithms that have been realized in hardware. Due to complexity problems, the dynamic thresholding algorithms that have been implemented have been simple and therefore do not detect thin faint lines. They also require parameter adjustment. The Laplacian algorithm is a more robust

algorithm that does not require much parameter adjusting. It detects fine details, but its main problem is that it detects and enhances all the noise also.

1.5. The IPT Scan Optimizer and the APS algorithm

Image Processing Technologies, Inc. (IPT) has developed a product known as the Scan Optimizer [IPT 1988] which performs the 8-bit to binary conversion accurately, robustly, and in real-time. This hardware system uses the Pseudo-Laplacian algorithm [Letellier et al. 1985], which is an extension of the Laplacian filtering. One set of parameters works well for a range of documents, and detects all the details that the Laplacian method does, but suppresses the noise. It is a complex algorithm, and for higher speed scanners it must perform 800 million operations per second.

Without parameter adjustment, the Scan Optimizer produces a superior image compared to other algorithms, but for its best possible image, the Scan Optimizer also requires fine tuning. This higher image quality is needed for improved OCR recognition rates and general readability. Archivists are demanding this quality and are now adjusting its parameters most of the time. This leaves us with one of our original problems, parameter adjusting.

An algorithm has been developed to set the Scan Optimizer parameters without human intervention. This is the Automatic Parameter Setting (APS) algorithm [Tatem 1990]. Until recently, this algorithm was implemented only in software, and was not capable of real-time operation. Specially designed hardware was needed for fast document conversion. The implementation of this hardware is the subject of this thesis.

1.6. Organization of Thesis

This thesis describes the hardware realization of the Automatic Parameter Setting (APS) algorithm for the IPT Scan Optimizer. The system has been designed, built, and tested with a large number of documents. Chapter 2 will briefly discuss the APS algorithm, the system requirements, and a high-level description of the final hardware. The remainder of the thesis will describe individual components of the APS hardware. Chapter 3 will describe how the different histograms are computed and Chapter 4 will describe how they are used to calculate the Scan Optimizer parameters. Chapter 5 will discuss the interface to the Scan Optimizer. Finally, Chapter 6 will demonstrate that the objectives have been reached, propose future developments, and summarize the APS system.

2. The APS Implementation

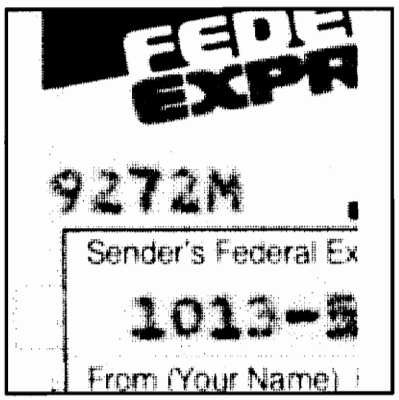
This chapter presents an overview of the Automatic Parameter Setter implementation. First, the fundamentals are covered by summarizing the algorithm which calculates the parameters for the IPT Scan Optimizer. The second section discusses the system requirements which were identified and addressed during hardware implementation. Finally, an overview of the completed hardware is given.

2.1. The APS algorithm

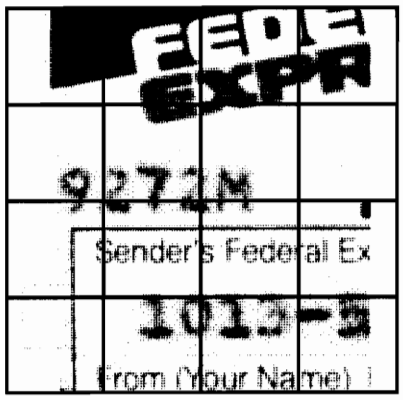
One key to the success of the APS algorithm is that the image is subdivided into smaller portions, and an optimal set of three parameters is found for each frame. Histograms are generated and analyzed separately for each subdivision of the image. The Scan Optimizer then converts each frame to binary form using that frame's particular settings. A full description of the APS algorithm was given in [Tatem 1990].

2.1.1. 64x64 Pixel Frames

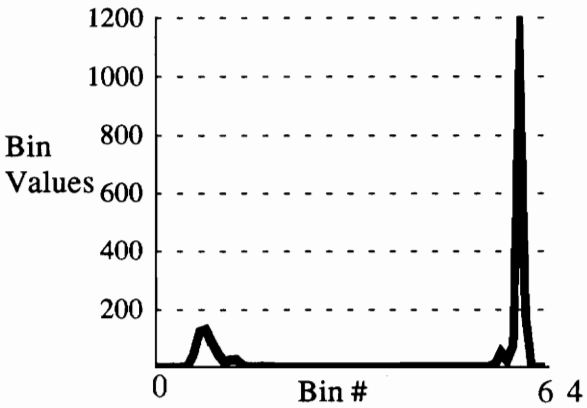
The selection of frame size is important for several reasons. If the frames are too small, not enough data will be contained in the frame, and the parameters may not be accurately adjusted for the data. If the frames are too large, the parameters may not be able to change rapidly enough for large changes in the document. Empirically, a good size was found to be a 64-x-64-pixel frame. Figure 2.1 (a) shows an original picture used and Figure 2.1 (b) shows an example of how a picture is subdivided.



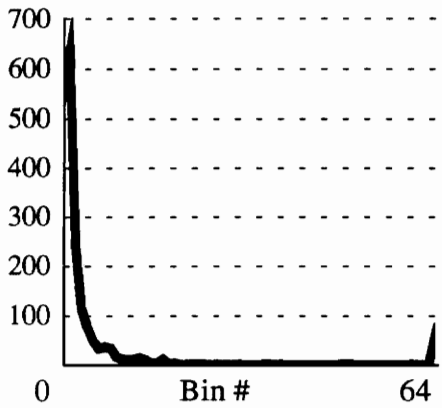
(a) Original Image



(b) Image Subdivided into Frames



(c) Gray Value Histogram
for Upper Left Frame



(d) Difference Histogram
for Upper Left Frame

(e) Definitions

Difference Width= width of low peak in difference histogram
InformationDiffVectors= the sum of the bins above the bin number equal to sensitivity
TotalDiffVectors= the sum of all the difference bins
GrayStart= the starting location of the gray value histogram
GrayEnd=the end location of the gray value histogram

(f) Formulas

Sensitivity =Difference Width / 3.5 + 1.5

Thickness= 32 * (informationDiffVectors / totalDiffVectors) + 0.5
Blackfill= (GrayEnd+GrayStart) / 2

(g) Parameters for Upper Left Frame

Sensitivity: 9
Thickness: 12
Blackfill: 73

Figure 2.1. Image subdivision and Scan Optimizer parameter calculation.

2.1.2. Histogram Generation

In the APS algorithm two histograms are calculated for a given frame. The first of these is a simple histogram of the 8-bit pixel values contained in the frame. The second of these is a histogram of the absolute values of the 8-bit differences between each pixel in the frame and its neighboring pixels. Examples of the two types of histograms are shown in Figure 2.1 (c) and Figure 2.1 (d).

2.1.3. Histogram Processing

For each image frame, the APS algorithm analyzes the shapes of the two histograms and then determines three parameters for that region. Ideally, each histogram contains two peaks. First, the height and width of the histogram peaks are found. Then the distance between the peaks is found. Using this information, a formula is then used to find the value of each parameter for the Scan Optimizer. Figure 2.1 (e) defines the different parts of the histograms that are used and Figure 2.1 (f) shows the formulas used for obtaining the parameters and an example calculation. This process is repeated for each in the image.

2.1.4. Scan Optimizer Processing

The parameters along with the 8-bit image data are then passed to the Scan Optimizer. The parameters are: *sensitivity* which relates to the level of change or edge that is considered significant and *thickness* which relates to the width of the edge required. These edges create outlines that need to be filled; this is performed by ORing the edges with a rough threshold of the intensity values from the original image. This threshold level is the *Blackfill* level. This information is processed through a complicated set of formulas [IPT 1988] and a binary image is produced which is sent back to the scanner.

2.2. System Requirements

This section describes hardware requirements that were identified at the beginning of the design process. Most of these have been met in the final implementation.

2.2.1. No Operator Intervention

The primary motivation for developing and implementing the APS algorithm was to reduce the need for operator intervention. The APS algorithm automatically fine tunes the parameters to optimum values, and the final hardware must faithfully implement this algorithm.

2.2.2. Image Results

The image results of the APS system connected to the Scan Optimizer should be comparable to the capabilities of the Scan Optimizer with manual fine-tuning. Tests performed in software have shown this to be true.

2.2.3. Speed

The APS system will be used in current Scan Optimizer installations. These systems are used for high-speed scanning of documents and film and typically operate at rates up to fifteen megapixels per second. In order to maintain these conversion rates, the APS system must be able to operate at similar speeds.

2.2.4. Flexibility

The ability to foresee and accommodate future design changes is desirable, but is a hard one to achieve during hardware realization. For example, future research could lead to an improved algorithm or new archiving problems could emerge. A goal in the design

of this system has been to use methodologies and components that will facilitate future changes.

2.2.5. Interface Requirements

This system will be used with the current Scan Optimizer in current installations. The APS system should therefore interface with existing physical connectors of the Scan Optimizer.

2.2.6. Cost

The APS system must be a viable commercial product. It is expected to save much time and effort by the end user, but affordability is an important design criterion. A target of $3000 was chosen as a maximum cost of parts.

2.3. APS Hardware Overview

2.3.1. Real-time Versus Near Real-time Processing

An important decision to be made early in the design process is whether the system will operate in *real-time* or *near real-time*. Real-time operation refers to a system which will accept input data without buffering and will generate output data at the same rate with virtually only a small delay; this usually requires design at the gate or state-machine level. A near real-time system is one which is fast, but does not quite meet the definition of the real-time system; these designs can usually be implemented with microprocessors and memory.

Two details greatly affect the possible hardware implementations. First, scanners generate large amounts of data. The largest page size for current Scan Optimizer

installations is 11"-x-17". The scan process transforms this to an image of 4,400-x-6,800 pixels at 400 dots per inch. To accommodate this and future demands, the Scan Optimizer has been designed to accept documents up to 32,768 pixels in width and of arbitrary length. The second important consideration is that the speed of today's scanners ranges from five to fifteen megapixels per second, and new scanners are expected to operate at twenty megapixels per second. The hardware implementation must be able to operate at comparable speeds.

2.3.1.1. Near Real-time Implementation

Memory Requirement

Current microprocessor systems are not capable of accepting input, calculating, and generating output at the scanner's pixel rate. To implement a microprocessor-based system, a frame buffer is required to temporarily store an entire image. This requirement means the system would not be able to realize the full potential of the Scan Optimizer.

To store a page of the minimum specified size, thirty megabytes of RAM would be required. This is prohibitive in cost ($1500 at current prices) and in board space. This expenditure might be justified if the RAM could be used for other purposes. For example, if a standard memory card could be purchased, design time would be reduced and the memory could be used by the computer for other applications when it was not in use by the image processing circuit. The scanner interface could be designed as a normal computer bus card that could communicate with the memory and the rest of the system. Unfortunately, this is not possible, because computer buses are not capable of sustained throughput at the top scanner speed (15 megahertz). The buses that are capable of such speeds, NUbus [Marshall et al. 1988], EISA bus [Glass 1989], and Microchannel [Cornejo et al. 1987], are not capable of sustaining it for three seconds. This implies that the

shared-memory concept is viable, but only if the memory card were redesigned with an internal scanner interface.

Speed possibilities

To obtain an acceptable speed, a very fast microprocessor is necessary. An example of this is a SPARC processor. SPARC processors and most other RISC (reduced instruction set computer) and DSP (digital signal processor) processors require one cycle time for internal instructions, two for read operations, and three for write operations. CISC (complex instruction set computer) processors such as the 80x86 family take longer [Fujitsu 1990, Intel 1990,AT&T 1990]. Assuming the near real-time system uses a Scan Optimizer, a rough estimate of the number of instruction cycles per pixel follows:

Cycles/ Pixel	Operation
	Data acquisition
0	Save data from scanner: done in real-time
	process histogram
2	read stored data
1	set new histogram address
2	read old histogram value
1	increment histogram value
3	store new histogram value
	Data Transmitting
3	to send to Scan Optimizer
	Loop
1	return back to the beginning of the loop
---	---
13 cycles per pixel	

This rough analysis indicates at least thirteen seconds in addition to scan time on a 32 megabyte image with a 30 megahertz RISC processor. The actual process time is most likely longer, because certain simplistic assumptions have been made. For example, no time has been added for parameter calculation or for initializing between calculations. These instructions are distributed over an entire frame, but could amount to one

instruction cycle per pixel. Another assumption is that these instruction times assume cache hits on every instruction or very high speed instruction and data memory [Fujitsu 1990].

Complexity Issues

The RISC processor system could possibly be purchased; if not, many known designs could be followed. Most of the complexity of this design is writing the high-speed software; the speed of the system depends on it. If it is not well written, it will still work, but slowly.

2.3.1.2. Real-Time Implementation

Memory Requirement

Because a real-time system processes data as soon as it becomes available, much less memory is required. The primary need for memory in the APS algorithm is a video delay buffer; this is used to synchronize the video input stream with the initial calculation of parameters. In this case, 64 image lines are needed to accumulate histogram information and an additional 64 lines are needed to calculate parameters. For an 11"-x-17" page, this implies a minimum need for six hundred kilobytes, or to utilize the full capability of the Scan Optimizer, four megabytes are needed.

Speed Possibility

Real-time hardware is possible using a highly pipelined digital design similar to that used in the Scan Optimizer. In the Scan Optimizer, Xilinx Logic Cell Arrays are used for most of the computing which allows the system to operate at scanner speeds up to fifteen megahertz. Xilinx arrays will be described in more detail in Chapter 3. Since the processing is performed during scanning, the extra time required for image processing is virtually zero.

Complexity Issues

To make a system highly pipelined, the design must be custom built for the application. In this design, Xilinx arrays would be used for the repetitive computing, and a processor or ALU of some sort for the more complicated arithmetic.

2.3.1.3. The Decision

After careful comparison of these two approaches, the real-time alternative was chosen. The deciding factors were the higher speed and reduced cost (because less memory is needed) as compared to the near real-time implementation.

2.3.2. Hardware Description

The resulting hardware fits on a double layer PC-AT card. Logically, the system is composed of three subsections: histogram generation, histogram processing and general interfacing. The last subsection includes scanner, Scan Optimizer, and host computer interfacing. Figure 2.2 shows the relation, among these sections.

2.3.2.1. Histogram Generation Section

The histogram generation section receives its input directly from the scanner A/D converter. Two histograms are generated: one of gray values and one of the difference values of the image. The resulting histograms are made available to the Histogram Processing section.

Histogram Generator Chip

Two identical circuits are used for the histogram generation. One of these circuits processes the video input data directly. The second accepts input from a difference generator chip. Both of these chips are Xilinx arrays. The histogram generator chip handles storage of the data in the RAM and controls the RISC processor's access to the data.

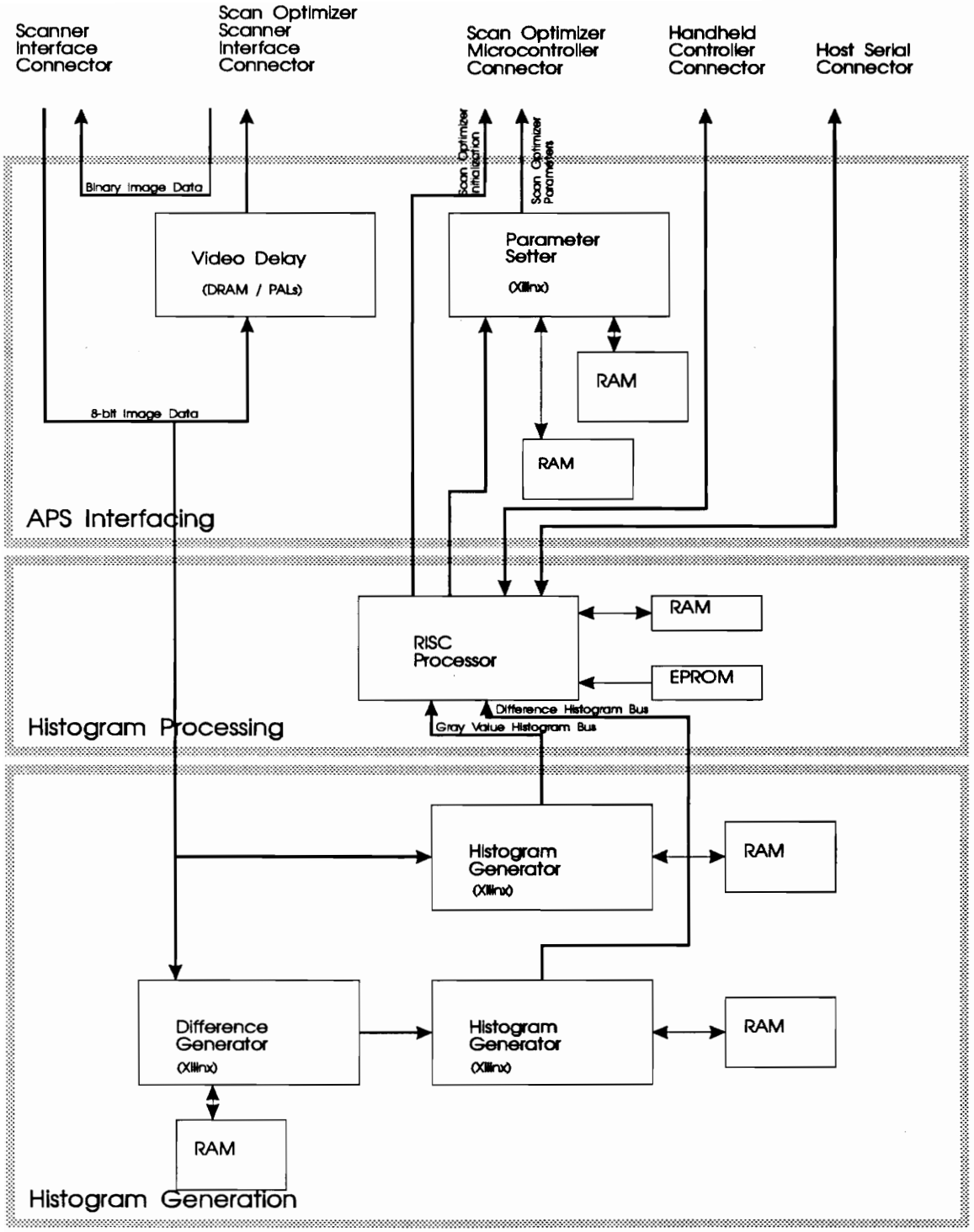


Figure 2.2. The APS hardware block diagram.

Data Flows:
Scanner Connector to Histogram Generation to Histogram Processing to APS Interfacing to Scan Optimizer Microcontroller Connector
Scanner Connector to Scan Optimizer Scanner Interface
Histogram Processing to APS Interfacing to Handheld Controller
Histogram Processing to APS Interfacing to Host Serial Connector

Difference Generator Chip

The difference generator, which is a Xilinx array, generates the necessary difference vectors for the difference histogram. It organizes which vectors it will send and when and sends the 8-bit data to the Histogram Generator chip that is designated for the difference histogram.

2.3.2.2. Histogram Processing

The histogram processing is handled by a Fujitsu RISC chip. Associated with it are interfaces to the other sections, a slow EPROM for boot, and a fast RAM for data and code.

2.3.2.3. APS Interfacing Section

The APS system is interfaced to the other systems by a set of connectors. The host access is a simple serial port, and the scanner interface connects to previously built interfaces for the Scan Optimizer. This interface however is a little more complicated.

The APS system is connected to the Scan Optimizer through two physical connectors, a scanner interface connector and a microcontroller connector. This section of the hardware requires two components to accomplish this, the video delay and the parameter setter chip.

Video Delay

The APS system is a pipelined system. The first set of parameters is computed after 128 image rows have been received. Thereafter, the APS produces a set of parameters for each 64 lines with no extra delay. These 128 rows can be passed to the Scan Optimizer only after the first parameter set has been computed. This initial 128 line delay is compensated in the video delay section which is a set of RAMs, latches and programmable array logic (PAL) chips.

Parameter Setter chip

Because the input image frames are changing every 64 pixels, a new set of parameters must be transmitted to the Scan Optimizer every 64 pixels. To reduce the overhead of this parameter setting, a separate hardware section composed of a Xilinx array and two RAMs is dedicated to this task.

3. Histogram Generation

The histogram generation section generates two histograms, a gray value histogram and a difference histogram. These histograms are calculated separately for each frame of the image and are stored in separate sections of RAM. These sections of RAM are then accessed by a RISC processor (described in Chapter 4) which analyzes them and determines the optimal sets of parameters.

To provide these operations, the histogram section receives inputs from two sources, the scanner interface and the RISC chip. The scanner interface sends 8-bit video input, a pixel clock, an end of line signal, and a frame clock which indicates the change of a frame; these signals are used to generate the histograms. The RISC chip provides a data access signal which indicates a desire to access the histogram data; the RISC clock is then stopped by a hold signal from the histogram generator until access is possible. The RISC chip also provides a data clock in order to time the receiving of the histogram data through a 12 bit data bus.

Since the system is real-time, histograms are also being computed and stored during the RISC access time. To prevent conflict between these two operations accessing the same memory, the new data must be stored in another location. After the set of histograms has been computed in one section, and the old values have been read from the other section, the RISC begins accessing the newly generated values.

3.1. Existing Systems

Two different types of hardware already exist to perform histogram generation. The first class consists of data acquisition boards such as one available from Amber

Engineering that incorporate additional circuitry for histogram calculations. These cost thousands of dollars and are therefore undesirable for this application. The second category contains commercial ICs such as one available from LSI Logic that is specifically designed for histogram generation; these cost from two to four hundred dollars. There are two main problems with both of these devices. The first is that they do not support simultaneous histogram calculation and access to the previously computed histogram. The second problem is that they do not support generating separate histograms for different frames.

With data acquisition boards, there is not much that can be done to address these problems, because these are existing boards that cannot be modified. With the histogram generator chip, however, a design might be possible. Separate circuitry would be necessary to control the RAM to fix the problems; this circuitry could be contained in a single Xilinx array that would cost sixty dollars. It was found that this same Xilinx array could perform the histogram computation at no additional cost. For this reason, the histogram processor IC alternative was not used.

3.2. Approximations

It is possible to generate accurate histograms in real-time, but analysis shows that approximations of these histograms are sufficient for Scan Optimizer parameter calculation and are simpler to implement. Since the purpose of a histogram is to distinguish the contents of an image frame from other frames, histogram contents can be substantially approximated and still represent the image frame well. Some of these approximations were considered in [Tatem 1990]; others were tested in software before

building hardware. Taking these approximations into consideration makes a smaller, faster histogram generation section possible.

3.2.1. Gray Histogram Approximations

A 64x64 frame contains 4096 pixels, and there are 256 possible values for each pixel. Most of the pixels values will cluster near one of two levels: the *background level* which is usually bright and corresponds to a large pixel value, or the *information level* which is usually dark and is represented by a pixel value near zero. The important information in the gray histogram that is used by the histogram processing section is the *distance* between these levels. Two approximations were made in the gray value histogram; the first will not affect this distance and the second will simply divide it by four.

The first approximation is to sample only every other pixel along the image row for the histogram. This reduces the total number of points in the histogram to 2048. In general, this merely reduces the height of the histogram by two. Since the height is not used in the gray value histogram, this approximation does not affect the histogram processing at all. Figure 3.1 (a) shows an original gray value histogram and Figure 3.1 (b) shows the effects of this approximation on the histogram.

The second approximation is to use 64 bins of pixel values instead of 256 bins. The result is as if the image were digitized with 6 bits per pixel. The distance between the histogram bins for the 8-bit case can be approximated by multiplying these distances by 4. There are two benefits of this. First, less memory is required. In addition, there are fewer bins to examine which reduces computation time. Figure 3.1 (c) shows the effects of the approximation on the histogram.

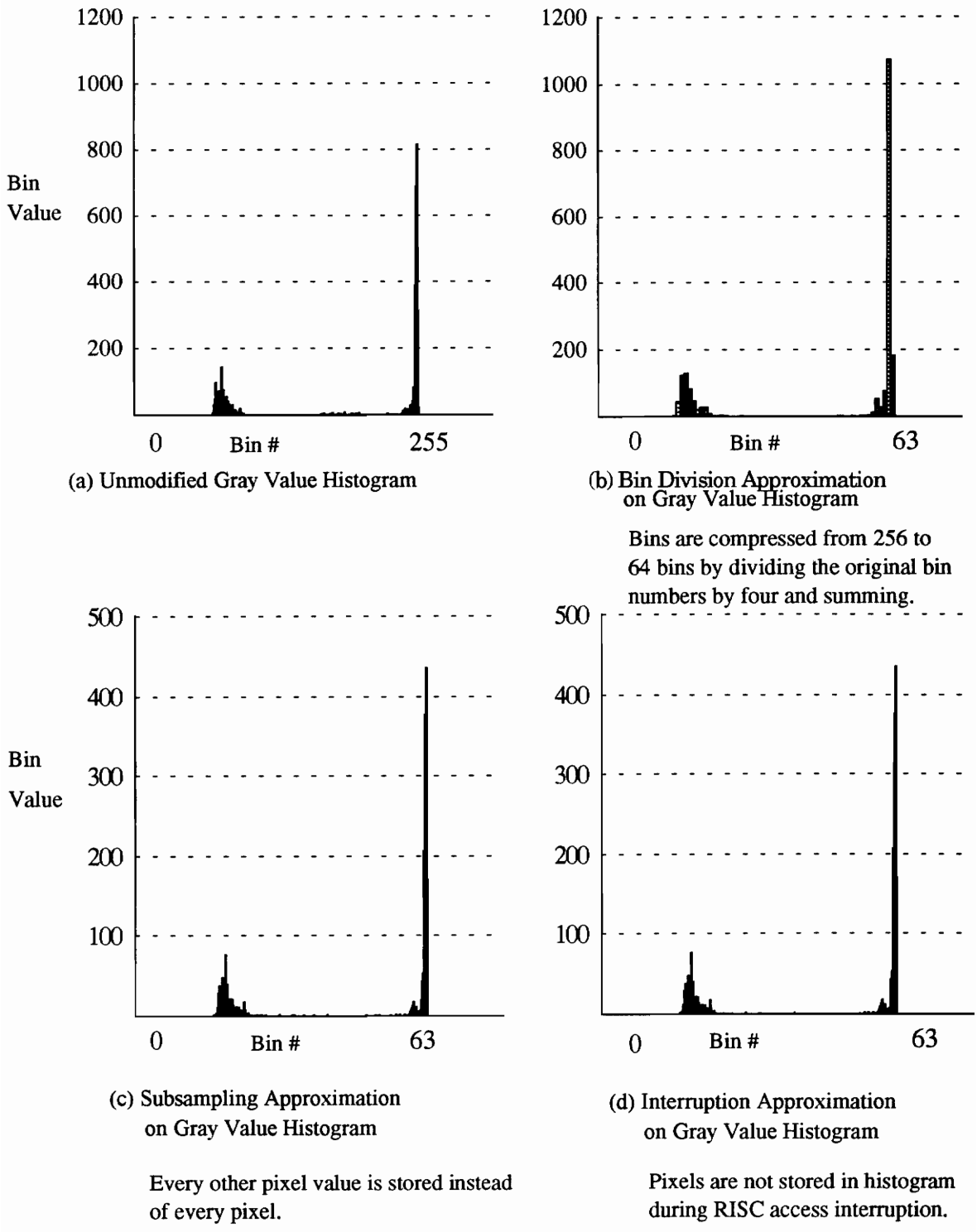


Figure 3.1. Approximations on the gray value histogram

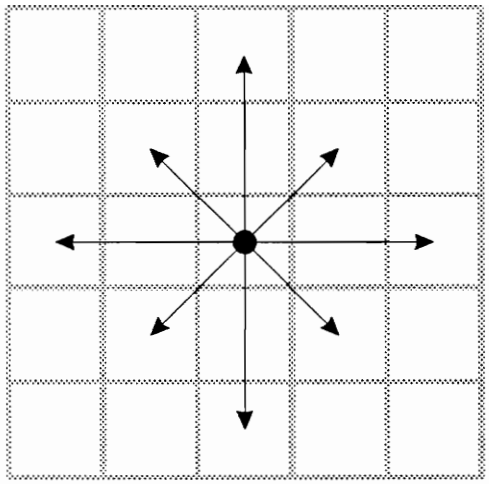
3.2.2. Difference Histogram Approximations

The difference histogram is used to observe the contrast ranges of the image. Because of this, the initial consideration for the difference histogram was to store all eight possible contrast values or differences for each pixel in the frame. However, this would be very difficult to perform in real-time since it would require eight values to be stored in the histogram every 70 nanoseconds. As a compromise, it was decided to compute only two vectors per two pixels and alternate at the set of vectors at the end of each line between North / South vectors and East / West vectors; the histogram is then calculated from this. Figure 3.2 shows the orientations of the difference vectors. Figure 3.3 (a) shows an original difference histogram and Figure 3.3 (b) shows the effects of the approximation on the histogram.

The number of histogram bins was reduced with the difference histogram, although a different method is used than with the gray value histogram. The important value used for parameter calculation for this histogram is the width of the lower histogram peak. There are two peaks, but the one used for calculation is near zero and its width is often less than 10; the width value needs to be very precise. Instead of compressing all of the 256 bins into 64 histogram bins, the resolution of the width value was maintained by placing all the difference values above 63 into a single saturation bin. Figure 3.3 (c) shows the effects of the approximation on the histogram.

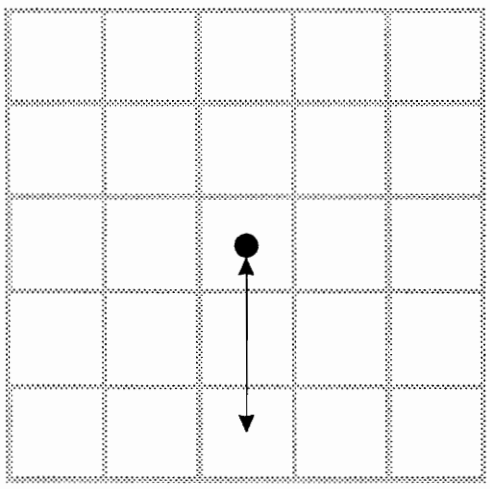
3.2.3. Multiple Access Interruption and Approximation

As mentioned in the beginning of the chapter, there is a possible conflict in memory access between the histogram generation and the RISC CPU. There are three possible approaches: provide separate memories for generation and access, use a multiport RAM, or give one device priority over the other for memory access. Since

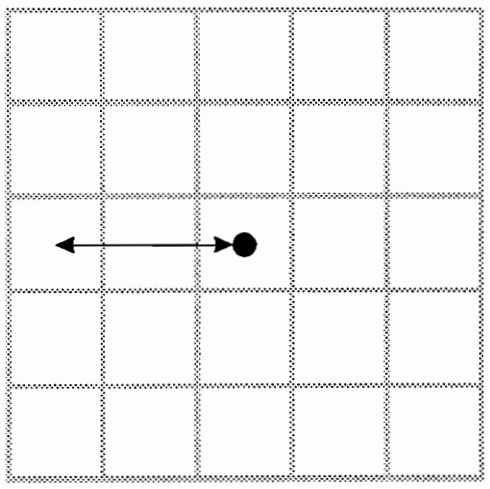


8 compass directions checked every pixel

(a) Ideal Difference Vectors for Checking Contrast



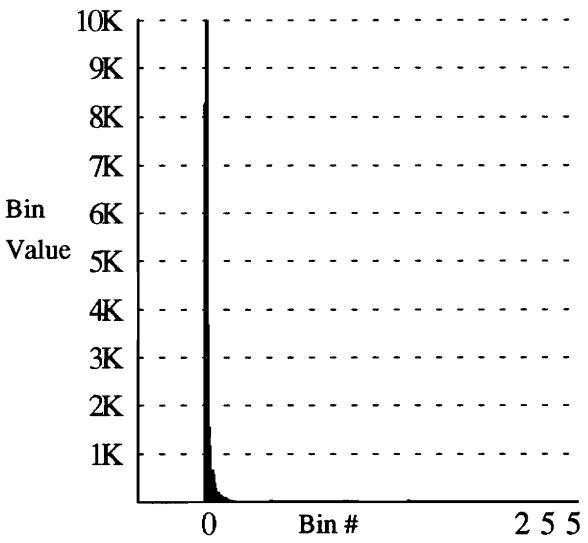
During Odd Lines: North and South vectors are computed every other pixel



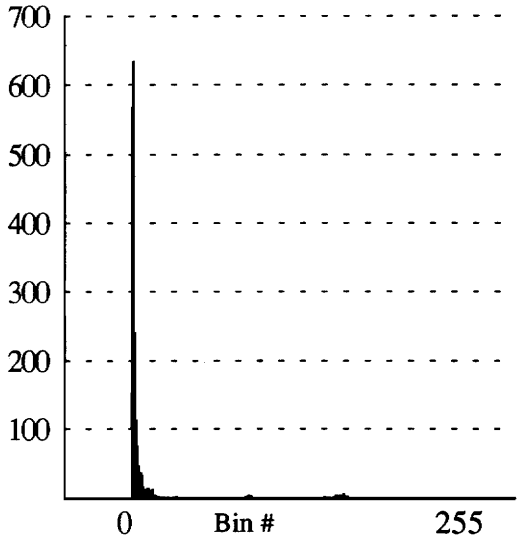
During Even Lines: East and West vectors are computed every other pixel

(b) Difference Vectors Used to Approximate Contrast

Figure 3.2. Map of difference vectors for checking contrast.

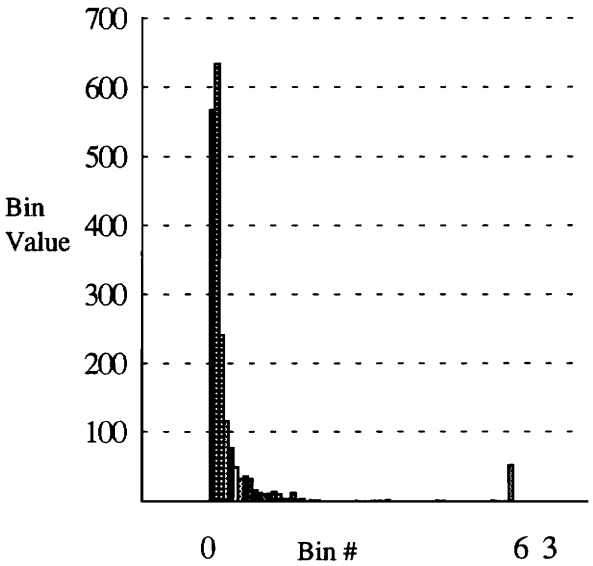


(a) Unmodified Difference Histogram



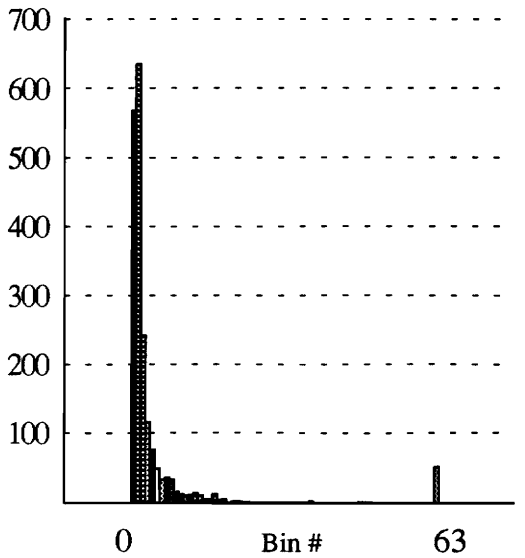
(b) Subsampling Approximation on Difference Histogram

Two difference vectors are stored every other pixel instead of eight vectors every pixel. (see figure 3.2)



(c) Saturation at 63 Approximation on Difference Histogram

256 bins are compressed into 64 by saturating the bins above 63.



(d) Interruption Approximation on Difference Approximation

Differences are not stored during RISC access interruption.

Figure 3.3: Approximations on the difference histogram.

approximations are allowable, the last method has been used because it is less expensive. Twice the amount of memory to store one bank of histograms is required, but the two banks can be contained in the same RAM device.

For this design the priority is given to the RISC chip. The RISC cannot wait long and cannot have loss of data while reading the histogram. Due to allowing approximations, the histogram calculation can be interrupted and can have a slight loss of data. This will mean that less than 2048 values are stored; this loss of data has been checked and the histogram is still represented correctly.

Since only 64 words must be retrieved from each frame, the RISC access will only cause approximately 64 values out of the 2048 to be lost. Figures 3.1 (d) and 3.3 (d) show the effects of this approximation on the gray value and difference histograms. Depending on when the RISC chip happened to access the saved data, more or less values may be lost. If a large amount of RISC access happens during one frame, too many histogram calculations may be interrupted. Then the histogram may not accurately represent the frame. This is unlikely due to the method of retrieval and calculation, however, a precaution has been placed in the RISC program to check for this. If too few histogram values are stored, the parameters for this problem frame will be approximated from adjacent frame data.

3.3. Xilinx Logic Cell Arrays for the Histogram Generation System

Xilinxes Logic Cell Arrays [Xilinx 1991] are used for most of the histogram generation. This provides many benefits: adaptability, reconfigurability, and high speed at

a low expense and small space. Appendix A has a brief description of Xilinx arrays. They enable the design to change easily, which would be useful if the APS system is ever upgraded. An example of some possible algorithm changes in the histogram generation section that could be accommodated are different frame sizes and different orientation of difference vectors.

3.4. Difference Subsection Hardware Realization

The difference subsection is shown in Figure 3.4. It is composed of a Xilinx chip and two banks of RAM. The banks of RAM are organized in series in order to provide three lines of video data to the difference Xilinx array. Inside the Xilinx array, a signal alternates at the end of each line to indicate if North / South or East / West differences are to be found. This signal switches the inputs to the subtracters. To compute the absolute value of the difference vector, two subtractions are done, one for each direction. The positive output is then selected and compared to 63; if it is found to be greater, it is saturated to 63. This final value is then sent out of the Xilinx chip to the Histogram generation chip.

3.5. Histogram Section Hardware Realization

Two histograms are generated and a single Xilinx chip is used for each. Histogram generation is a much more involved process than it initially appears to be, especially since each update must be performed in 140 nanoseconds. In addition to performing this, the chip must also handle the RISC access arbitration. Figure 3.5 shows a block diagram of this section.

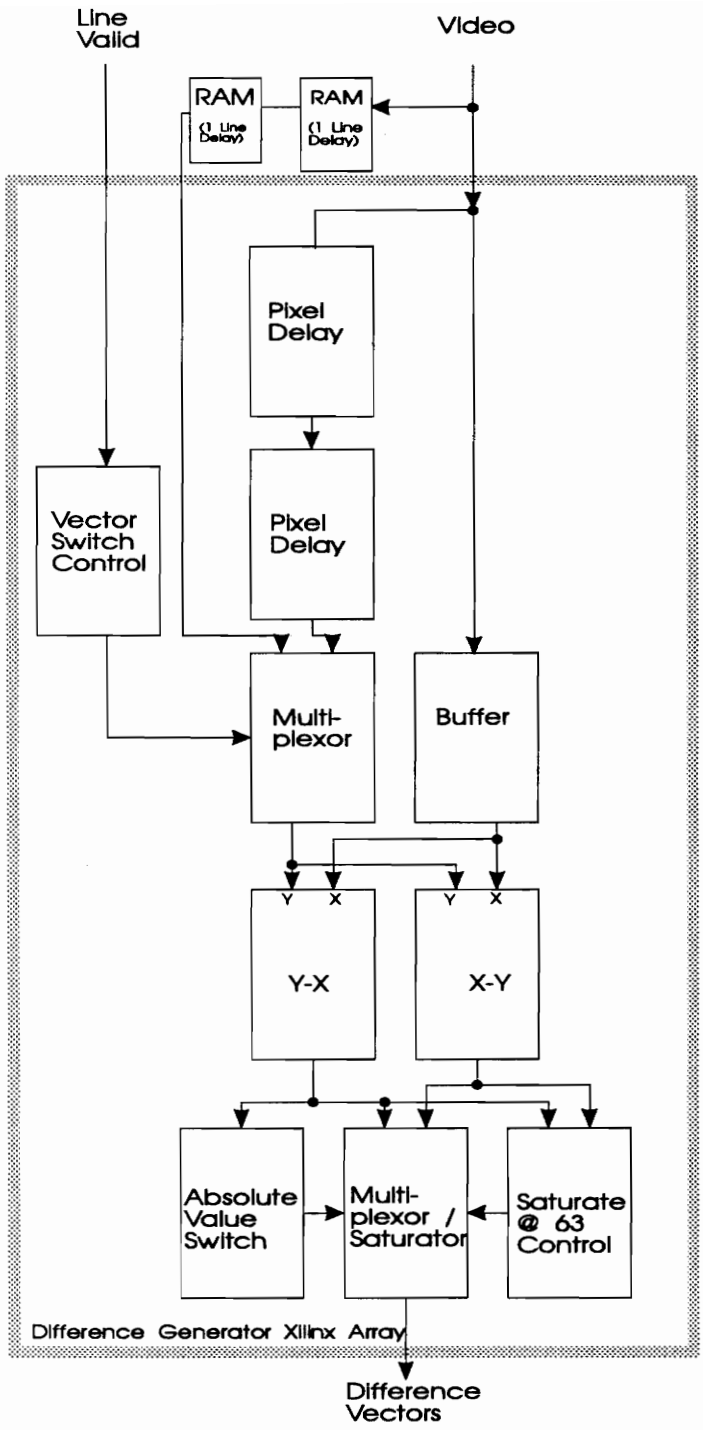


Figure 3.4. Difference generating part of the histogram generating section.

An output is generated every other pixel
During Odd Lines: the positive north / south vector is output
During Even Lines: the positive East / West vector is output
The output is then saturated at 63

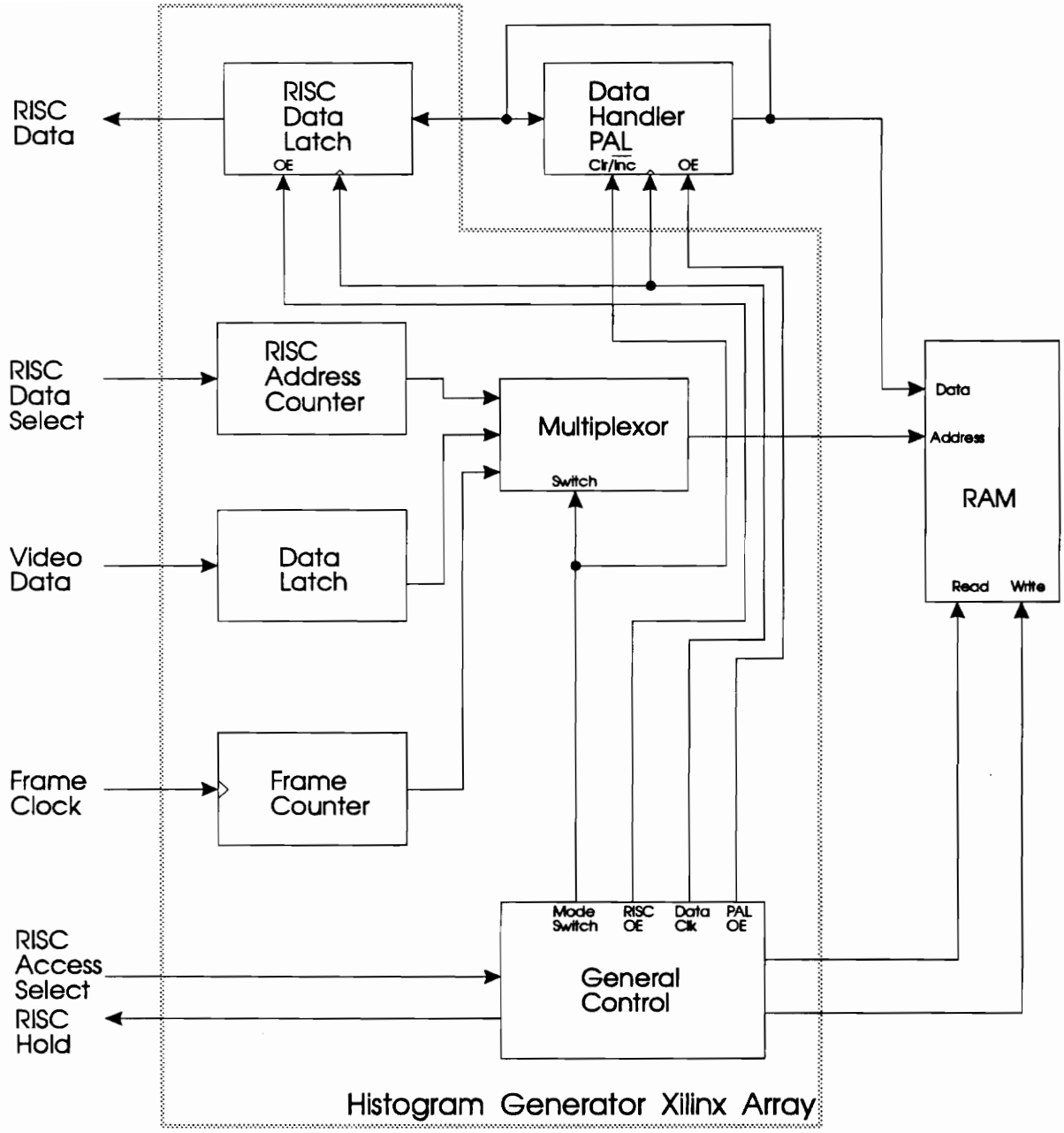


Figure 3.5. Histogram generation section.

Data Flows:
Video Data/ Frame clock to multiplexor to Ram Address -
Ram data (read) to Pal (increment) to Ram data (store)
RISC Access select to general control to RISC hold
RISC data select to counter to Multiplexor to RAM address -
RAM Data to RISC data - PAL (zero output) to RAM data (write)

3.5.1. Processes

For histogram calculation, the process is as follows:

1. receive new half speed scanner clock
2. latch pixel value
3. generate address for particular frame and pixel value
4. read old value
5. increment
6. store new value

For the RISC access, the process is as follows:

1. receive RISC histogram access signal
2. hold processor clock until current histogram calculation is finished
3. receive RISC data clock
4. generate address for the histogram location read
5. read value and send to RISC
6. store zero in that location
7. wait for next RISC clock or for RISC histogram access signal to go inactive

3.5.2. Timing

Chip delays are similar in Xilinx to those in VLSI technology. Internally, routing and gates have very low delays, but signals are delayed greatly by exiting the chip; this is due to delay caused by input/output buffers. Read-modify-writes are performed for both Histogram calculation and RISC access. This causes problems since signals must exit or enter the Xilinx chip three times. The timing delays are as follows:

10 ns: output buffer delay
25 ns: RAM read access time
10 ns: input buffer delay
30 ns: computation time
20 ns: arbitration multiplexing control
10 ns: routing delays
10 ns: output buffer delay
25 ns: RAM write access time
total = 160 ns (only 140 ns is available)

Since the total is more than available, a significant problem has been found. To solve this problem, an external PAL was used. This PAL accomplishes the input/output buffering, computation, arbitration multiplexing, and routing in 25 ns. Signals then only need to exit or enter the chip once per operation which reduces overhead. For RISC

access, only reading is required, and for Histogram calculation, only writing is required. The Xilinx chip is still necessary due to the many inputs, multiplexing and complicated timing control.

3.5.3. Histogram Generator Xilinx Array

In the Histogram Generator Xilinx array, there are basically two modes of operation, the RISC access mode and the histogram calculator access mode. Address generation for both modes is performed in the Xilinx array and is multiplexed to the outside RAM depending on the mode. The Xilinx array also arbitrates the switching of modes for the internal controls, the controls to the RAM, and the PAL. The majority of the data control is handled by the PAL, and only direction signals are sent to the PAL to direct its actions.

The histogram calculation access address generation is complicated. There are 64 possible bins in each histogram, and there is a histogram calculated for each frame across the image. The generated RAM address points to a specific bin of a specific frame; the last 6 bits corresponds to the bin number which is equal to the input histogram data, and the remaining address bits are equal to the frame number. The frame number is reset at the beginning of the line and is incremented at the end of each frame which is indicated by the frame clock; this happens every 64 scanner clocks.

The RISC access address generation is a fairly simple operation. Histogram values are stored sequentially, and a counter is used to step through these addresses. At the end of the last row frame, when the banks of memory are switched, a counter is cleared. Every time the RISC accesses the Histogram generator, the RISC sends a data clock. This clock simply increments the counter.

The first responsibility of the control section is to generate RAM and PAL signals for histogram calculation. While in the histogram calculation mode, the RAM address is multiplexed appropriately and the PAL is signaled to increment. For each data value, the RAM is signaled to read, and the PAL latches the RAM data, the RAM is signaled to write and the PAL output enable is activated. The new incremented histogram value is then written to the RAM.

The next responsibility of the control section is to handle the switching to RISC access. The signal RISC Data Select comes from the RISC and indicates when it desires to access the histograms. At that point, a hold signal which holds the RISC clock goes high until the current video data has been added to the histogram. The hold signal is then disabled which lets the clock continue, and the mode is switched to RISC access.

During RISC access, the control section generates a different set of signals for the RAM and PAL. First the RAM is placed in read mode, and the data is latched in the Xilinx array. Next the RAM is placed in write mode, and the PAL is enabled to output all zeros to the RAM data lines; this clears the histogram bin. Then when the new histogram is generated for the next row of frames, the bins will start at zero.

3.5.4. Data Incrementing and Clearing PAL

The PAL receives a 12-bit histogram value from the RAM and must either increment or clear that value for writing back to RAM. This operation is switched by an additional PAL input signal, the Clr/Inc signal. This requires the PAL to have 13 inputs and 12 outputs. The operations are performed by a fairly simple but lengthy set of equations and is shown in the appendix in the PAL description; the most complicated equation is for the highest bit which requires 12 sum terms. Since the data lines are bidirectional and go to both the inputs and outputs, the PAL must also be able to latch the

inputs and have a separate output enable. The most critical design decision was to find a PAL that could accommodate all the requirements; the best suited chip found was the AMD26V12.

4. Histogram Processing

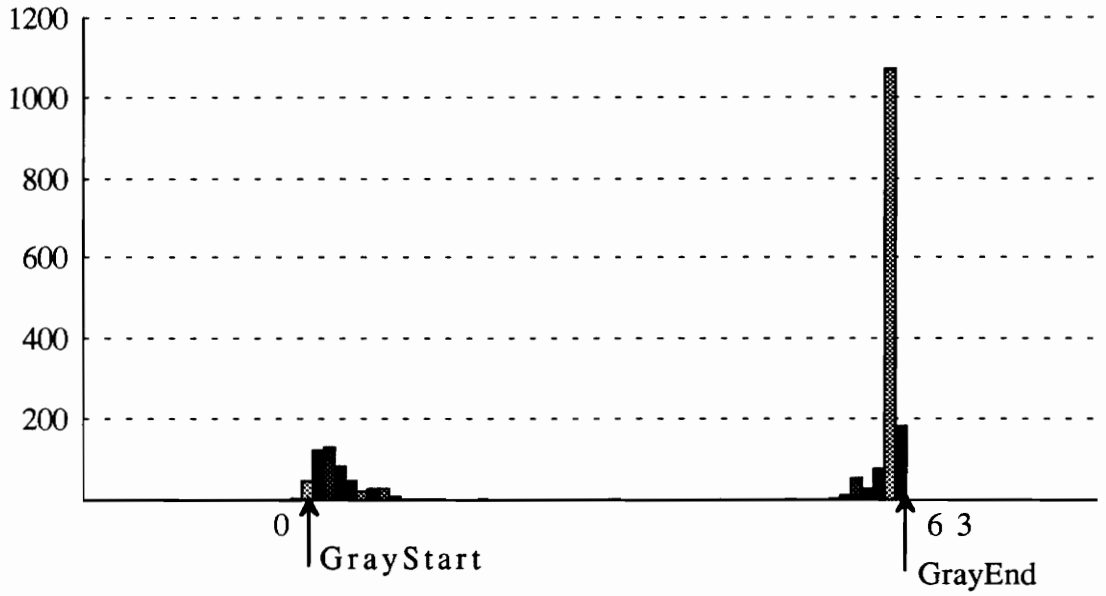
Histogram Processing is performed by a RISC processor with supporting hardware. Histograms are read from the Histogram Generation section, they are analyzed, and the resulting optimal set of parameters is sent to the Scan Optimizer Interface section which performs the actual setting. This chapter describes this software procedure, the approximations made, and the hardware designed to implement the section.

4.1. Histogram Analysis

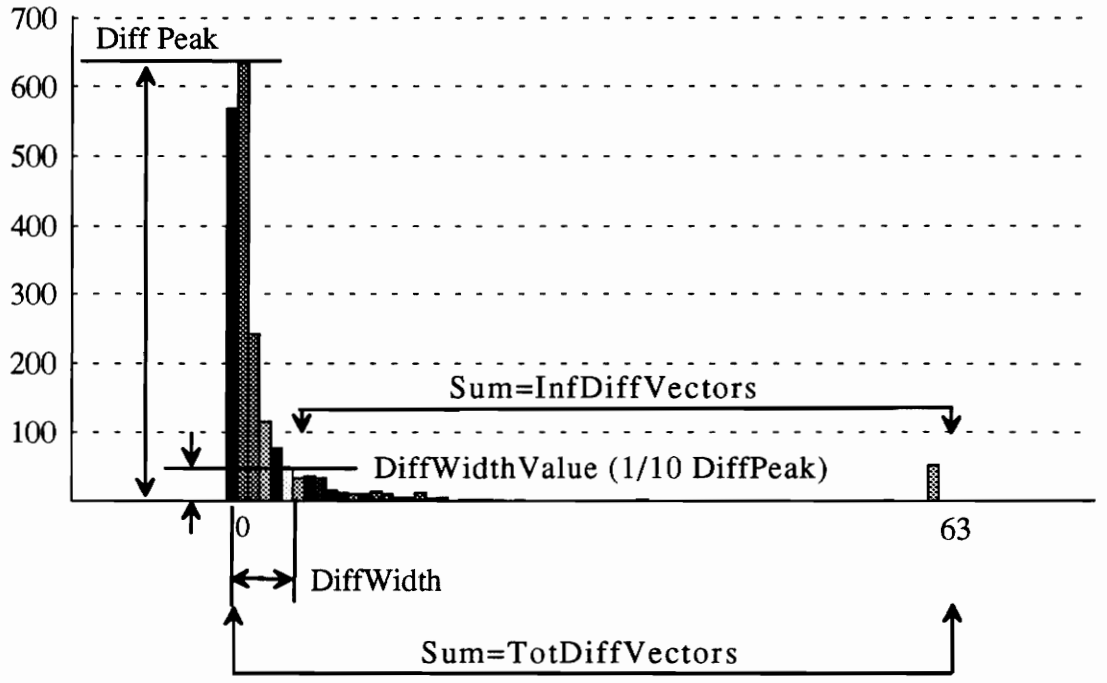
Analyzing histograms is complicated due to the many ways to describe the shape. For most documents, there are at least two peaks. For calculating the Scan Optimizer parameters, it is useful to calculate the distance between these peaks and the height and width of each peak. The slopes of the peaks may also be important. Quantifying these values can be very difficult since histograms are generally not smooth and have poorly defined peaks. For the algorithm used here, a single analysis method with simple formulas was formed to approximate multiple analysis and more complex formulas. Figure 4.1 graphically shows the parts of the histograms used, and the pseudocode for the analysis procedure and a discussion follows:

4.1.1. Bimodal Assumption

Certain assumptions have been made in the formulation of the method of processing. These assumptions will simplify things and speed the process. The first assumption is that the document will have bimodal histograms. For normal text



(a) Gray Value Histogram With Important Values Marked



(b) Difference Histogram with Important Values Marked

Figure 4.1. Histogram analysis values.

documents this is mostly true. For the gray value histogram, there is usually a single peak for the background level and a single peak for the information level. For the difference histogram, there is usually a large peak near zero for the background noise and a smaller peak at a greater value for the difference between the information and background levels.

If there are distinct multiple background or information levels, a bimodal histogram is not generated. For most of these cases, there still is only a single background. It may be a noisy or fading one which covers a wide range of values, but usually it is one peak. As long as this is true, the most important computation of finding the level of background noise is still being made correctly, thereby making the overall parameter selection close to optimal.

4.1.2. Difference Histogram Width and Sensitivity Calculation

The lower peak width is necessary for sensitivity calculation, but first the height must be found. The height calculation is approximated by examining the location of the desired peak and finding the maximum value; in this case the lower histogram values are checked. As the area is scanned, the maximum bin value and bin location is kept.

The peak width is defined to be the location of one-tenth the maximum height. This is approximated by starting from the location of a maximum height and proceeding forward in the histogram bins until the bin values descend to one-tenth the maximum height; this is the right edge of the peak. This calculation is being performed on the lower difference histogram peak which always starts at zero; therefore the left edge does not need to be found, and the width of the peak is equal to the bin number of the right edge.

The sensitivity value is defined to be as follows:

$$Sensitivity = \frac{Difwidth}{3.5} + 1$$

Since the Fujitsu SPARC chip cannot do division, this operation becomes a little more complicated. Speed is extremely important, therefore, a lookup table was created.

This was the fastest division method possible, and it does not take up too much memory space. Because sensitivity is an integer parameter, this value is then truncated to an integer value.

The pseudocode for the sensitivity calculation follows:

```
; find low difference histogram peak
maxvalue = 0
for binnum = 0 to lastbin / 4 do
begin
  if ( difhist[ binnum ] > = maxvalue )
  begin
    maxvalue = difhist[ binnum ]
    maxloc = binnum
  end
end

; find low difference histogram width
for binnum = 0 to lastbin do
begin
  if ( difhist[ binnum ] < 0.1 * maxvalue and difhist[ binnum + 1 ] < 0.1 * maxvalue
) break
end
difwidth = binnum

;CALCULATE sensitivity
sensitivity = difwidth/3.5 + 1.5
```

4.1.3. Number of Difference Vectors and Thickness Calculation

The thickness parameter varies according to the level of background noise. To find the amount of background noise the difference histogram is analyzed and the vectors associated with information, ie. those greater than the sensitivity value, are counted and compared with the total number of vectors. The number of information difference vectors was found by summing the difference histogram bins at and above the sensitivity value. The total number of difference vectors was found by summing the difference histogram bins below the sensitivity value and adding it to the number of information vectors.

These two values are then used to find the thickness parameters as follows:

$$Thickness = 32 \times \frac{InfDiffVectors}{TotDiffVectors} +$$

As for the sensitivity calcuation, this operation is performed by using a lookup table. The thickness parameter is also an integer parameter, so this value is truncated to an integer value.

The pseudo code for the calculation of the thickness parameter follows:

```
;find number of information difference vectors
infdiffvectors = 0
for binnum = sensitivity to 63
begin
  infdiffvectors = infdiffvectors + difhist[binnum]
end

;find total number of difference vectors which includes background noise
totdiffvectors = infdiffvectors
for binnum = 0 to sensitivity -1
begin
  totdiffvectors = totdiffvectors + difhist[binnum]
end

;CALCULATE thickness
thickness = int( 32 * infdiffvectors ) / totaldiffvectors )
```

4.1.4. Gray Value Histogram Starting and Ending Location and Blackfill Calculation

The start and end locations of the gray value histogram are needed for calculating the blackfill level. These values are found by starting at the correct end (left for the start and right for the end) and proceeding to the other side. Sometimes there is speckle noise, so a histogram start or end is not declared by the first non-zero bin, but by the first bin having a value greater than two.

The blackfill level is defined as follows:

$$Blackfill = \frac{GrayStart + GrayEn}{2}$$

The pseudocode for the blackfill calculation follows:

```
;find gray histogram starting point and lo peak location and height
graystart = -1
graylo = 0
graylovalue = 0
```

```
for binnum = 0 to lastbin / 2 do
begin
  if ( graystart = -1 and grayhist[ binnum ] > 2 ) graystart = binnum
  if ( grayhist[ binnum ] > graylovalue )
  begin
    graylo = binnum
    graylovalue = grayhist[ binnum ]
  end
end

;CALCULATE blackfill
blackfill = (graystart + greyend) / 2
```

4.1.5. Document Type Switches

After experimenting with many documents, certain documents were found to need different sets of formulas; a good example type of document is a blueprint. Most documents have background noise which is of fairly low in contrast when compared to the contrast of the lines in the document. Blueprints, however, often have high contrast background noise. Currently, a switch on the handheld controller has been made to toggle between the two types of documents, normal and blueprint. Since blueprints are usually archived by themselves and not with other documents, this should not be a problem, and the scanner should still be able to operate in batch mode. In the future, automatic detection of the document type will probably be possible.

4.2. Hardware Realization

The hardware implementation of the Histogram Processing section must be very fast and low in cost. To accomplish this, an integer RISC chip with high speed RAM is used. In addition to histogram processing, interfaces to the different APS sections, to the Scan Optimizer for initialization, and to the host computer must be implemented. Figure 4.2 shows a block diagram of the Histogram Processing section.

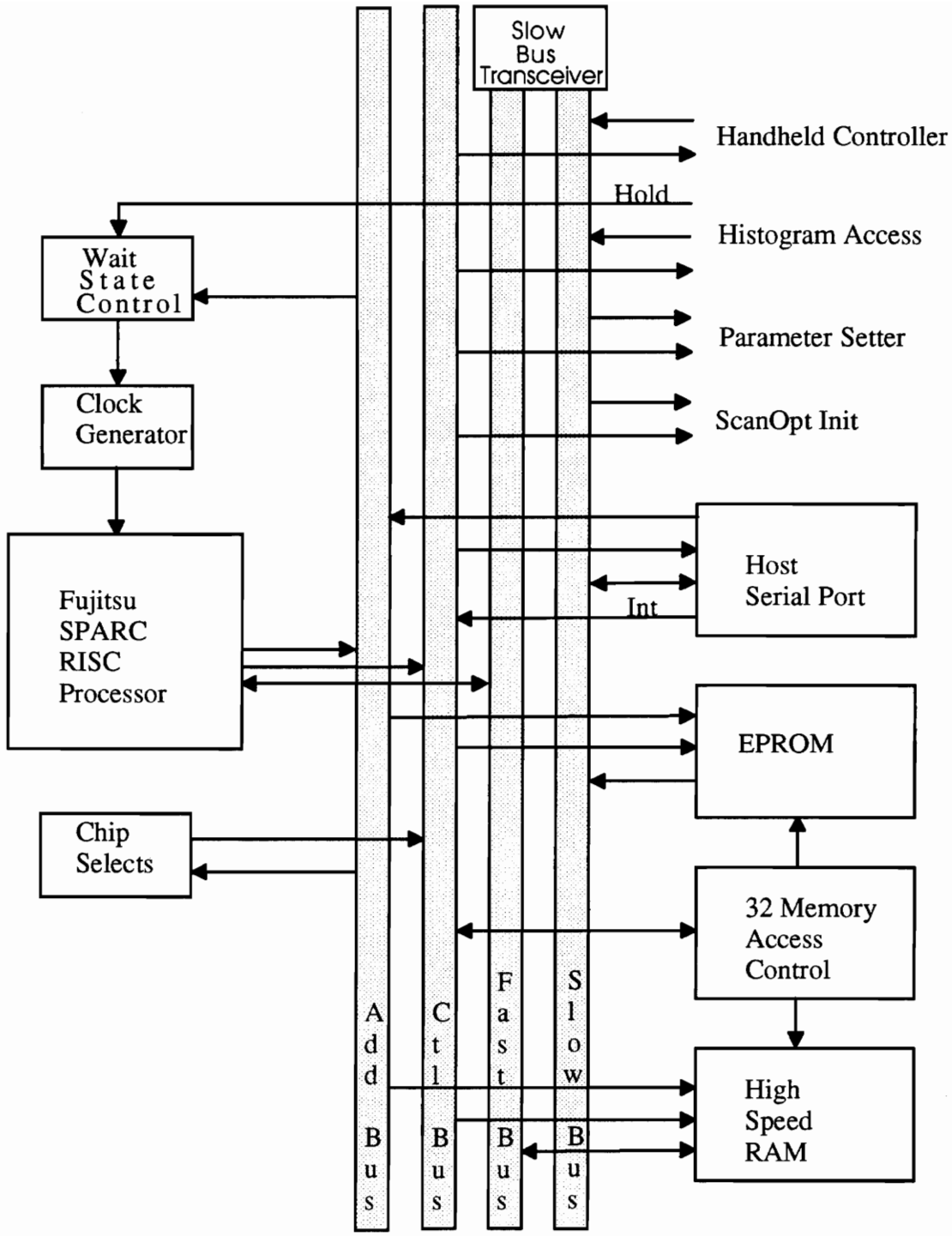


Figure 4.2. Histogram processing block diagram.

4.2.1. Processor

A number of alternatives were considered for the main processor. In particular, we considered the following architectures: microcontroller, CISC, RISC, and DSP. An integer RISC was chosen for its speed, ease of programming, and low cost.

Most of the required operations involved large amounts of simple data manipulation. Because CISC processors are slow at accessing data, and microcontrollers are slow in general, these two alternatives were rejected.

That leaves the RISC and DSP options. Both types of processors often have calculation sections for real numbers; however this increases the chip's price but not the speed for this application. DSPs usually also have special instructions for signal processing, but these special instructions are not needed for this algorithm. This special instruction set can make it difficult to program in assembly language which will be necessary for maximum speed.

For these reasons, several RISC processors were carefully considered. Ultimately, the highest speed (25 MHz) Fujistu SPARC was chosen. At the time, this was one of the fastest and least expensive integer RISC chips available. This product has integer arithmetic instructions only, and can perform fast I/O.

4.2.2. Memory Requirements

Two types of memories are needed by the RISC chip: EPROM and RAM. An EPROM is used to store the program for the RISC, but later, high-speed processing is executed from RAM. Due to the EPROM's slow speed, usually over 100 nanoseconds access time, the execution of all code from here would be entirely too slow. For the RISC chip to operate at its top speed, the RISC chip needs a memory device with an access time of 20 nanoseconds. This would be expensive, if not impossible, to have with an EPROM.

However, static RAMs can store the time-critical portions of code at a fairly inexpensive price. This code can be stored permanently in the EPROM but can be loaded and run from the static RAM. The only problem is that storage capacity of a RAM is very low. Therefore, only the high speed histogram processing code is to be loaded into the high speed RAM.

Most of the system devices (described below) can be accessed by only 8 out of the 32 bits of the data bus. This has been done to reduce board complexity. However, the RISC chip expects all program code to be 32-bit accessible, so all 32 bits are connected to the RAMs and EPROMs. A PAL was also required to control 8-bit access out of the 32 bits; this operation is done by simply selecting separate 8-bit wide memory devices according to the appropriate control signals.

4.2.3. Memory Space Allocation

The devices are grouped together into four locations according to speed. These locations are: high-speed RAM (0 wait states), Scan Optimizer Initialization, Histogram, and Parameter setting access (3 wait states), EPROM (5 wait states), and the serial chip (7 wait states).

The clock generation, wait states, and control bus are controlled by a set of PALs. Two PALs are used for clock generation and wait state control, two more for chip selects, and one more for RAM/EPROM 32-bit access controls (described in the section above).

4.2.3.1. Wait States

Since many of the devices connected to the RISC chip are slow and require multiple clock cycles to access, a wait state system had to be implemented. With the Fujitsu RISC chip, there are two methods of introducing wait states: halting the processor clock, or a hold signal. The hold signal method requires more complicated state machine

operation, whereas the clock halting requires more complicated timing. Since a high speed PAL was being used for implementing the two phase clock, high speed PALs were also used here to implement the clock halting method.

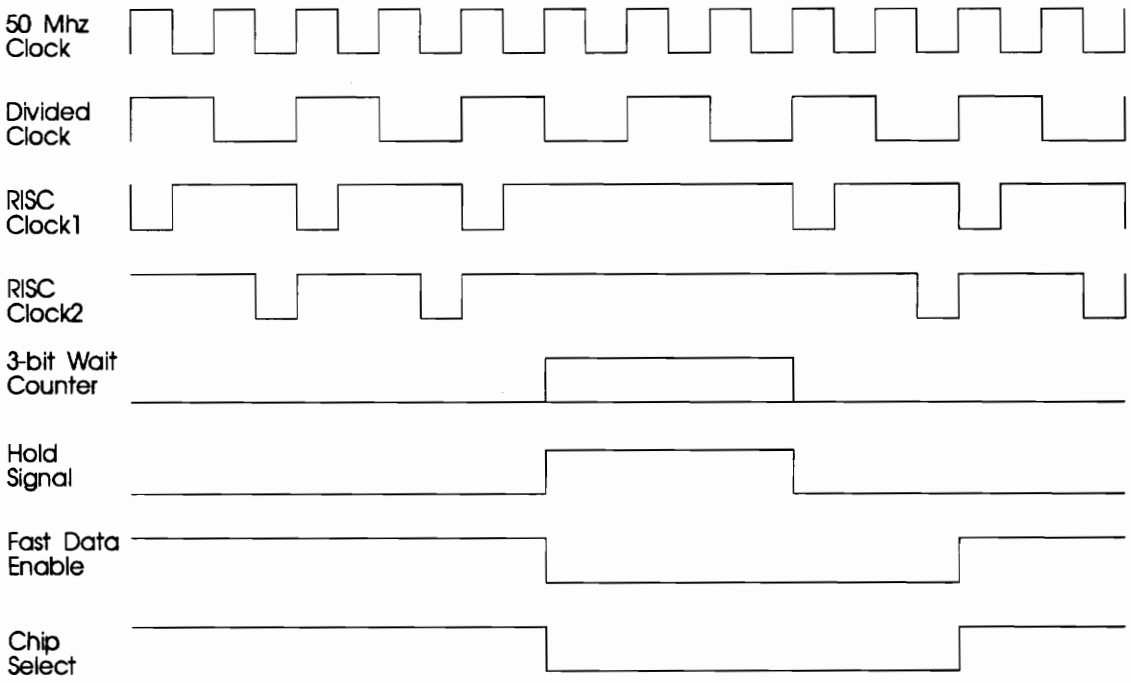
Two clocks, CLOCK1 and CLOCK2, are used to clock the RISC chip; CLOCK1 is high for the first 25 percent of the cycle and CLOCK2 is high for the last 25 percent of the cycle. For adding wait states to the system, the clocks are stopped when they are both low during the second and third quarters of their cycles. After the appropriate amount of time is passed, the clocks resume their normal operation. Two PALs control this clock halting operation. One PAL is used to count the wait state cycles and control the read and write signals, while the other PAL controls the CLOCK1 and CLOCK2 signals. Figure 4.3 shows the relation of these signals and PALs.

4.2.4. Fast and Slow Buses

Since such a contrast in speed exists between the fast devices and slow devices, two separate data buses were used. One fast bus operates at the highest speed possible, and assumes virtually no bus latency (the speed at which devices tristate); only the static RAMs use this bus. The slower bus operates at the speed of the slowest device in the system in terms of bus latency. The fast bus is connected directly to the RISC chip, but there is a bank of transceivers between the RISC chip and the slow bus in order to isolate the bus latency problem to the slow devices. Two signal for the direction and output enable are decoded from the address, read, and write signals by one of the PALs.

4.2.5. Interrupts

Two interrupts are used in the APS system. One is for the serial port, and one is used to start histogram processing. The serial interrupt is connected to the serial interface chip and is used to signal that newly received data is available or that the transmit buffer is



RISC clocks are generated from the combination of the 50 MHz and the divided Clock.

Wait State Operation

1. A specific address and a read or write signal triggers a wait state counter.
2. A hold signal then becomes active until the Wait counter reaches 0.
3. This hold signal halts the RISC clock and triggers the following signals.
4. Fast Data Enable and Chip select stay active until the next RISC clock.
   Fast Data Enable allows the transfer of data between the fast and slow data bus.
   Chip Select for the desired chip goes low which activates it.

Figure 4.3. Wait state and clock generation.

empty. The histogram processing interrupt is set up to force the processing to start anew when the new histogram data appears. If polling were used and processing was not completed in time, the end computations would use the wrong data, the output parameters would be erroneous, the system would be set out of sync, and the entire rest of the image would have incorrect parameters. With an interrupt, if the entire processing is not finished in time, the right side of the page will be set with previous parameters which may be incorrect, but the left side of the page will be correct. This simplifies debugging and yields a better image if there are any problems.

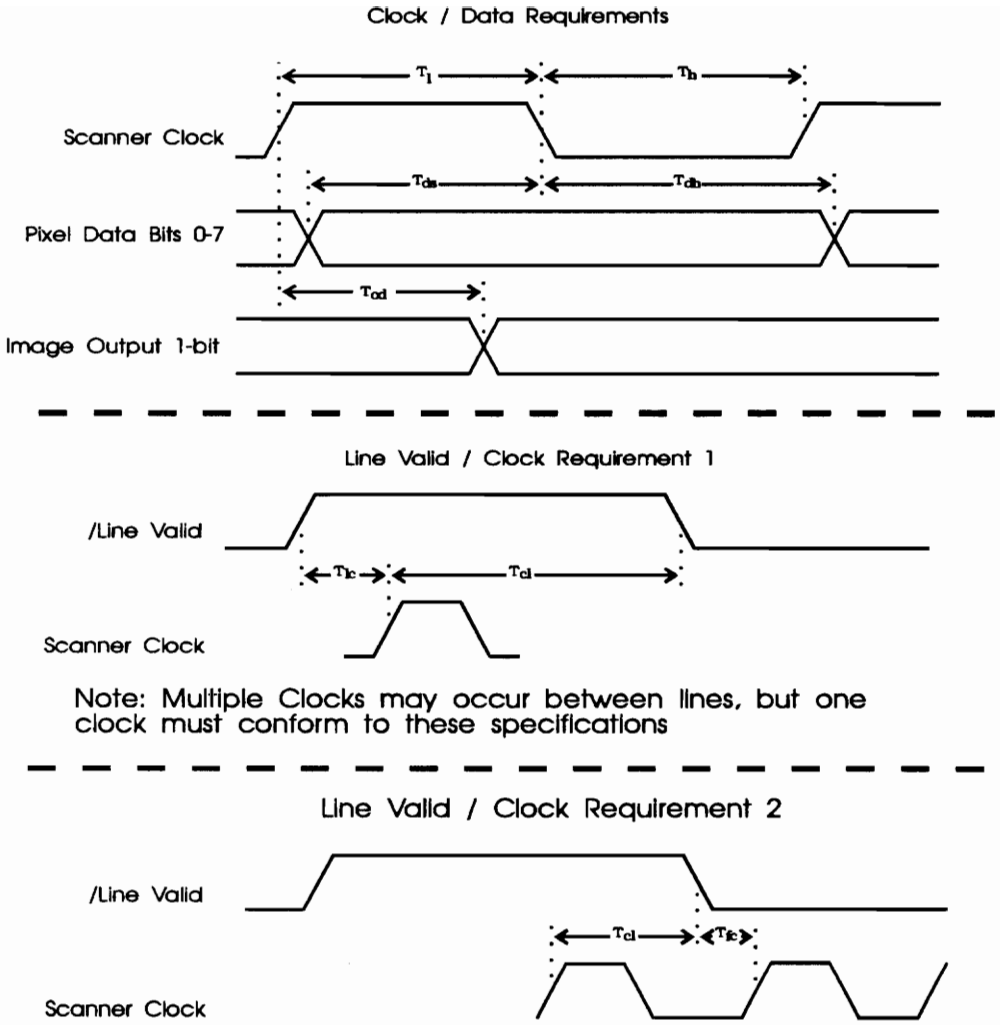
5. APS Interfacing

The APS hardware connects to four different devices: the scanner, a host computer, a handheld controller, and the Scan Optimizer. The latter is controlled through two ports, the Scan Optimizer Scanner Interface Connector and the Scan Optimizer Microcontroller Connector. The first three connectors are described in the following sections. The Scan Optimizer connectors and their associated circuitry are then described. The Scan Optimizer Scanner Interface Connector requires a video delay buffer, and the Scan Optimizer Microcontroller Connector requires a special parameter setting system.

5.1. The Scanner Interface

The APS system uses the same scanner interface boards that the Scan Optimizer normally uses without the APS. This way all the scanner interface boards already made can be used with the APS system. This interface connector is well documented in the IPT Scan Optimizer interface documentation [IPT 1991].

Most importantly, the APS system receives 8-bit gray scale data for each pixel from the scanner at this connector. Pixel values are received sequentially for each line of the image. In addition to the data stream, there are control signals: a clock signal to distinguish between adjacent pixels, and a line valid signal that distinguishes the end of one line and the beginning of another. The APS system generates one output signal for the resulting binary image, and sends this back to the scanner as a bit stream. Timing requirements are defined for these inputs and output which are shown in Figure 5.1.



Timing Values

Symbol	Min	Max	Units
Tl	35	-	ns
Th	35	-	ns
Tds	2	-	ns
Tdh	27	-	ns
Tod	20	53	ns
Tlc	12	-	ns
Tcl	25	-	ns
Tfc	12	-	ns

Figure 5.1. Scan Optimizer scanner interface signal requirements.

5.2. The Handheld Controller Interface

Normal operation does not require parameter changes. However, an option is available for document inversion and for the special case of blueprint documents. These options are available on a handheld controller connected to an input buffer which is received by the RISC processor.

5.3. The Host Computer Interface

A port is provided for full serial communication with the host computer. This allows minimal control similar to the handheld controller and more control for other uses such as controlling the Scan Optimizer as though the APS system was not connected. In the future, other special host computer requests might be desired. This type of communication can be given best through a serial port. This port is connected directly to the RISC processor which controls the rest of the APS system.

5.4. The Scan Optimizer Scanner Interface Connector

The Scan Optimizer Scanner Interface Connector is connected to where the Scanner Interface board plugged into the original Scan Optimizer. Now, these Scan Optimizer signals will come from the APS system, instead of the interface board.

Most signals which pass from the APS system to the Scan Optimizer can remain unchanged from the Scanner Interface Board. These scanner control signals can remain

the same and the output of the Scan Optimizer can be passed straight back to the scanner. By passing the signals straight through, all the necessary signal timing for the scanner and Scan Optimizer will be complied with. However, as we stated in a previous chapter, the input video data may not be passed straight through; it must be delayed by 128 lines.

5.5. The Video Delay Buffer

The sequence of inital processing events inside the APS system are as follows:

During Video Input Row 0:
    Histograms are generated for Row 0.
During Video Input Row 1:
    Histograms are generated for Row 1.
    Parameters are calculated for Row 0.
During Video Input Row 2:
    Histograms are generated for Row 2.
    Parameters are calculated for Row 1.
    The Scan Optimizer processes the image for Row 0.
During Video Input Row 3:
    Histograms are generated for Row 3.
    Parameters are calculated for Row 2.
    The Scan Optimizer processes the image for Row 1.
During Video Input Row 4:
    Histograms are generated for Row 4.
    Parameters are calculated for Row 3.
    The Scan Optimizer processes the image for Row 2.

The system has a lag of two rows between video input and Scan Optimizer processing. This implies the need to provide 128 lines of video delay in order to synchronize this initial parameter calculation delay with the incoming video data. In order to keep the video delay compatible with the Scan Optimizer's 32 kilobyte line lengths, 4 megabytes of delay are necessary, and a throughput of one pixel every 70 nanoseconds is needed.

5.5.1. Dynamic over Static Memory

There are two types of memory possible for the video delay, dynamic and static RAM. There are advantages and disadvantages for each. Dynamic memory is inexpensive and is available in large sizes, but is slower and needs special refresh circuitry. Static memory is fast and requires no refreshing, but is expensive and is available only in smaller sizes. Ultimately the deciding factor was price; a 1 megabit static memory costs around $60 whereas a 1 megabit dynamic memory costs around $6. For the entire video delay memory, this translates to roughly $200 versus $2000 in parts cost. Therefore, a strong incentive existed for using dynamic memories.

5.5.2. Hardware Realization

5.5.2.1. Column refresh

Dynamic memories are arranged in a square array that is accessed by a row and column address; every cell in a 64 kilobit memory has an 8-bit row address and an 8-bit column address. Due to the nature of these cells, each row needs to be accessed every 10 milliseconds in order to retain the row cells' contents. This required accessing process is called refreshing. Usually a counting circuit is designed to count the rows and the time elapsed, and then each row is accessed at the appropriate time.

The main problem is designing a refresh which does not interrupt normal memory access. Data is being input to the video delay at a sustained rate of around 15 Megahertz. If the refresh is allowed to interrupt this data flow, complications occur. First of all, a high-speed FIFO must be used during the interruption, and then afterward, the data must be sent at an even higher rate to the RAM. This prevents this from being an easy solution.

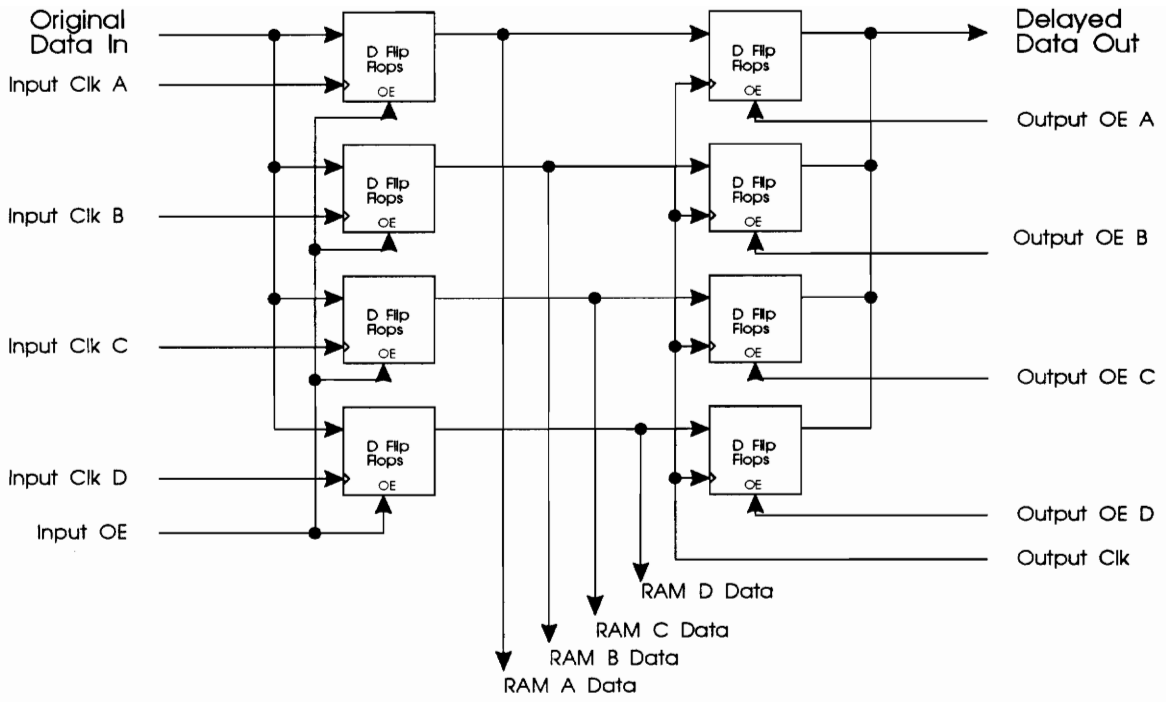
There is, however, an easy out; the memory is being cycled through completely. This fact lets us use the address of the RAM for counting the refresh row. The normal

address access is not quite enough, unfortunately; the entire memory is accessed in approximately a third of a second which is more than the required refresh rate. The solution is to switch the row and column addresses of the memory; the rows will then be cycled through much faster, approximately in 0.12 milliseconds. This method accesses every row in the required amount of time, and removes the interruption problem.

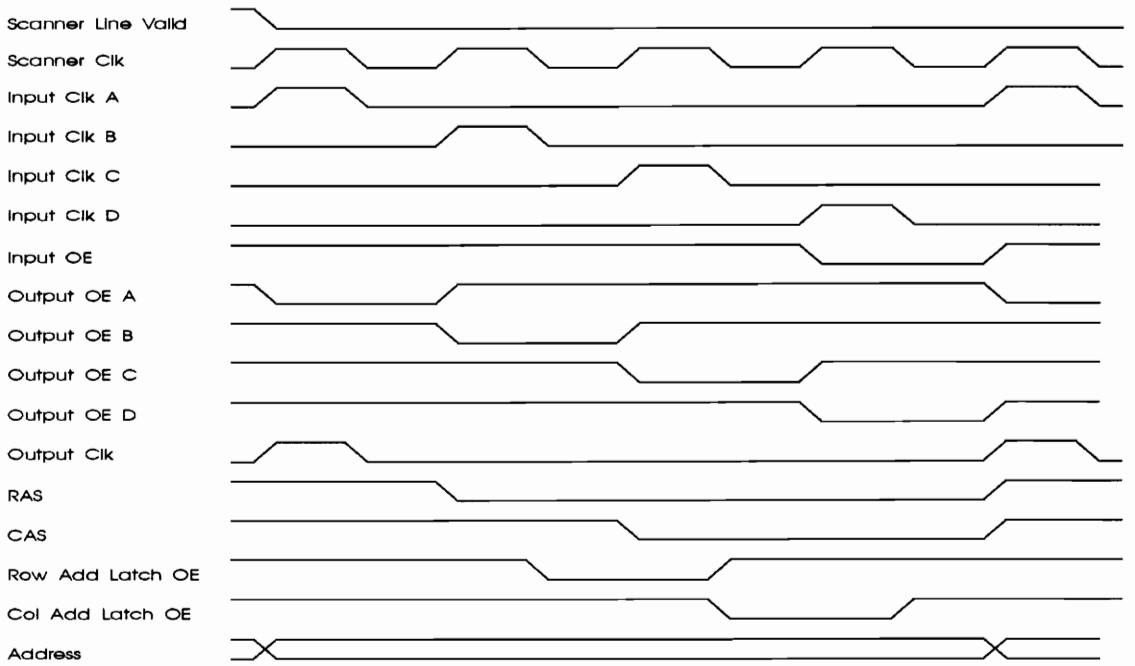
5.5.2.2. Speed and Interleaving Process

Speed is still a problem. The operation being performed in the RAM is a Read-Modify-Write cycle which in a high speed dynamic RAM can take approximately 190 nanoseconds [Hitachi 90]. To achieve the storage rate of a byte every 70 nanoseconds, interleaving is required. A bank of latches is designed to read and write four bytes of data at the same time.

To perform the interleaving efficiently, many control signals are needed and must be carefully timed. Four separate clocks and one output enable must be generated for the input latches, and one clock and four output enables need to be generated for the output latches. Row and column address strobes and read and write pulses are necessary for the RAM. Since using one clock to generate all the signals easily creates timing conflicts, using a higher speed clock was considered. New problems are related to this, however; the new clock would have to be synchronized with the scanner clock. Instead of this approach, a scheme was devised to use both phases of the clock for four cycles to Read-Modify-Write four bytes of data simultaneously. This control sequence is shown in Figure 5.2.



(a) Interleaving Control Block Diagram



(b) Interleaving Control Timing

Figure 5.2: Video delay interleaving DRAM control.

OE = Output Enable
Clk = Clock

RAS = DRAM Row Address Strobe
CAS = DRAM Column Address Strobe

5.6. The Scan Optimizer Microcontroller Connector

When the APS is not used, the Scan Optimizer contains a 8031 microcontroller board which initializes the Xilinx chips, sets the Scan Optimizer parameters in the Xilinx chips, and interfaces to a serial port for host communication. Since there is a RISC processor on the APS system which can perform these functions, the microcontroller board is no longer necessary.

The Xilinx array initialization will be performed identically to the old microcontroller board which is described in the Xilinx databook [Xilinx 1991]. The Xilinx chips are placed in a "Slave" initialization mode, and two signals are used for configuration, one for clock and one for data.

The method of parameter setting in the old Scan Optimizer system also uses two signals, one for clock and one for data. Including scanner dependent parameters and others which are constant, the 30 bits of parameters must be serially sent to the Scan Optimizer. Since this method configures parameters at a slow rate between pages, this is a problem. In the APS system, the parameters must change quickly between each frame in the middle of a page. Therefore, modifications in the Scan Optimizer and a parameter setting chip will be necessary.

5.7. The Parameter Setter

There were three major considerations in this circuitry. First of all, the original Scan Optimizer needed to be redesigned for high-speed parameter changes. Next, the problem of changing thresholds in the middle of pipeline processing was to be addressed,

and lastly, the actual parameter setter chip which performs the high-speed changes was designed.

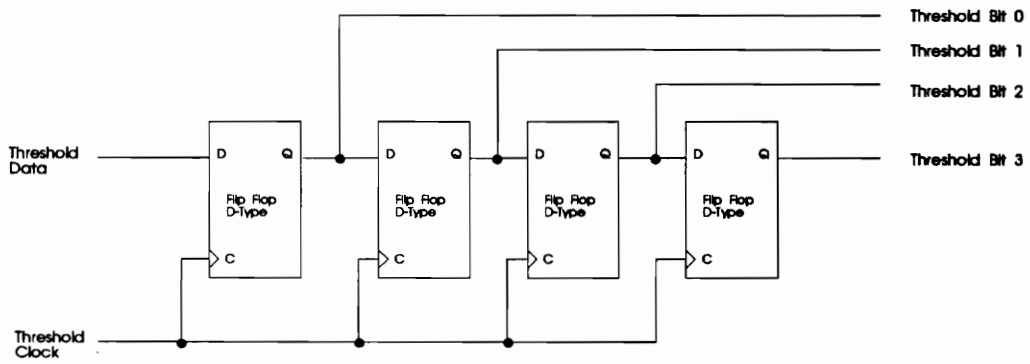
5.7.1. Revising the Scan Optimizer Parameter Setting Procedure

The Scan Optimizer was designed for a single setting of parameters per scan that would be changed manually. This meant that the parameters would be changed slowly between pages. The Scan Optimizer was designed so that a change in parameters takes about 45 microseconds. This is obviously too slow for the APS system which needs to change parameters between two pixels, which could be as fast as 70 nanoseconds.

The parameters are all stored in Xilinx chips; an example of 4-bit parameter storage is shown in Figure 5.3 (a). The internal blocks are configured as a simple shift register. Since the circuit is in Xilinx chip it can easily be changed to fix the problem; this is shown in Figure 5.3 (b). The basic change is that there are two parameter buffers: a temporary shift register and a new parameter buffer. The shift register buffer is a temporary buffer for loading slowly with the current method. When it is completely loaded, it is transferred in parallel to the new parameter buffer. Since this operation must happen quickly and in careful synchronization with the scanner clock, the parallel loading signal is only a clock enable which enables the scanner clock to cause the parallel loading. In the end, the parameter setting control to the Scan Optimizer has remained the same except for one extra signal that enables the parallel loading to occur.

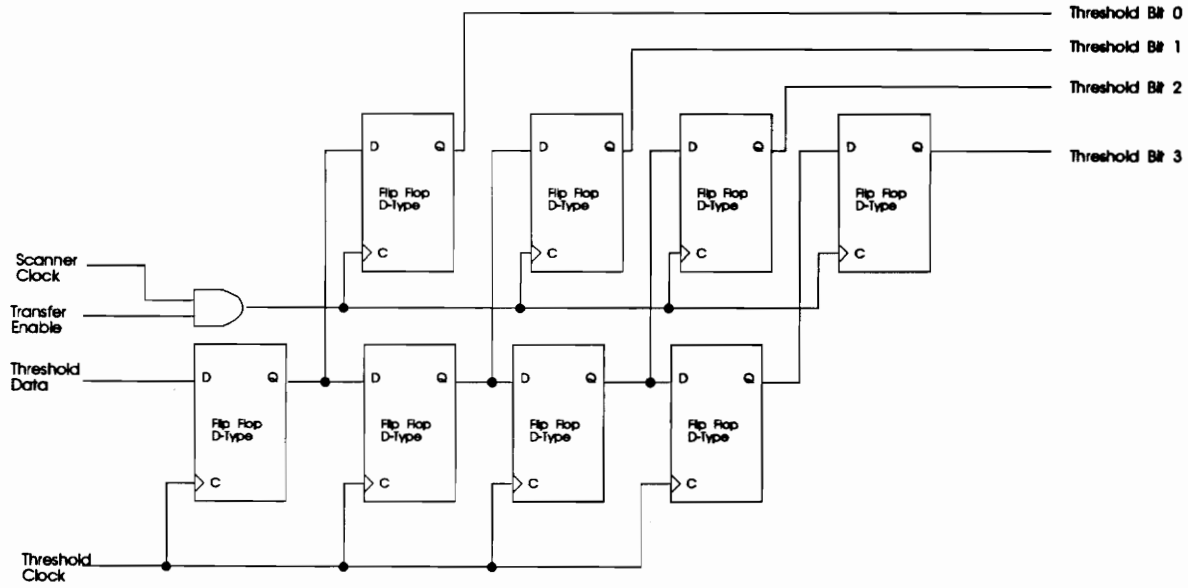
5.7.2. Pipeline / Threshold Changing Problem

Ordinarily when thresholds are changed in a pipeline, the pipeline must be flushed; the parameters are changed, and those values that were flushed must be started though the pipeline again. Since this is not possible in a real-time system when parameters are changing, a major problem seems to have appeared.



Old Xilinx Threshold Input Method

Designed for slow threshold input
Designed for minimal space
Uses single bank of flip flops
- This bank is serial loaded



New Xilinx Threshold Input Method

Designed for Fast Threshold Input
Uses dual bank of flip flops
- One bank for serial input
- One bank for fast input
- Parallel shift is performed between the two banks

Figure 5.3. Xilinx threshold input methods.

Surprisingly, in the case of the APS system, this problem can be ignored. The aim of the APS system is to vary close to one setting for the entire page, therefore, a minimal number of similar settings are used for a page. The fact that the settings are all similar, and the fact that the Scan Optimizer is a robust system, cause no visual artifacts in most cases. The only case where problems can be seen is where drastic parameter changes do occur. This only occurs in problematic areas where the Scan Optimizer usually can only pick up noise. In these cases, the noise in one frame may cause a problem called tiling effect, but the interior information is more discernable than before.

5.7.3. Xilinx Chip for the Parameter Setter

A Xilinx chip with accompanying RAM was chosen for the parameter setting process. It once again offers adaptability, reconfigurability, and high speed at a low expense and small space.

Two algorithm changes could happen later that can easily be accommodated for. These are changing the frame size and the number of parameters sent to the Scan Optimizer. The Xilinx array's flexibility for changes allows this to be changed fairly easily, thus not limiting the design to its original concept.

5.7.4. Hardware Realization

There are two tasks the Parameter setter chip must perform. First of all is the actual setting of the parameters in the Scan Optimizer. Secondly, it must receive and store the parameters from the RISC chip. Figure 5.4 shows this in a block diagram of the Parameter setter subsection.

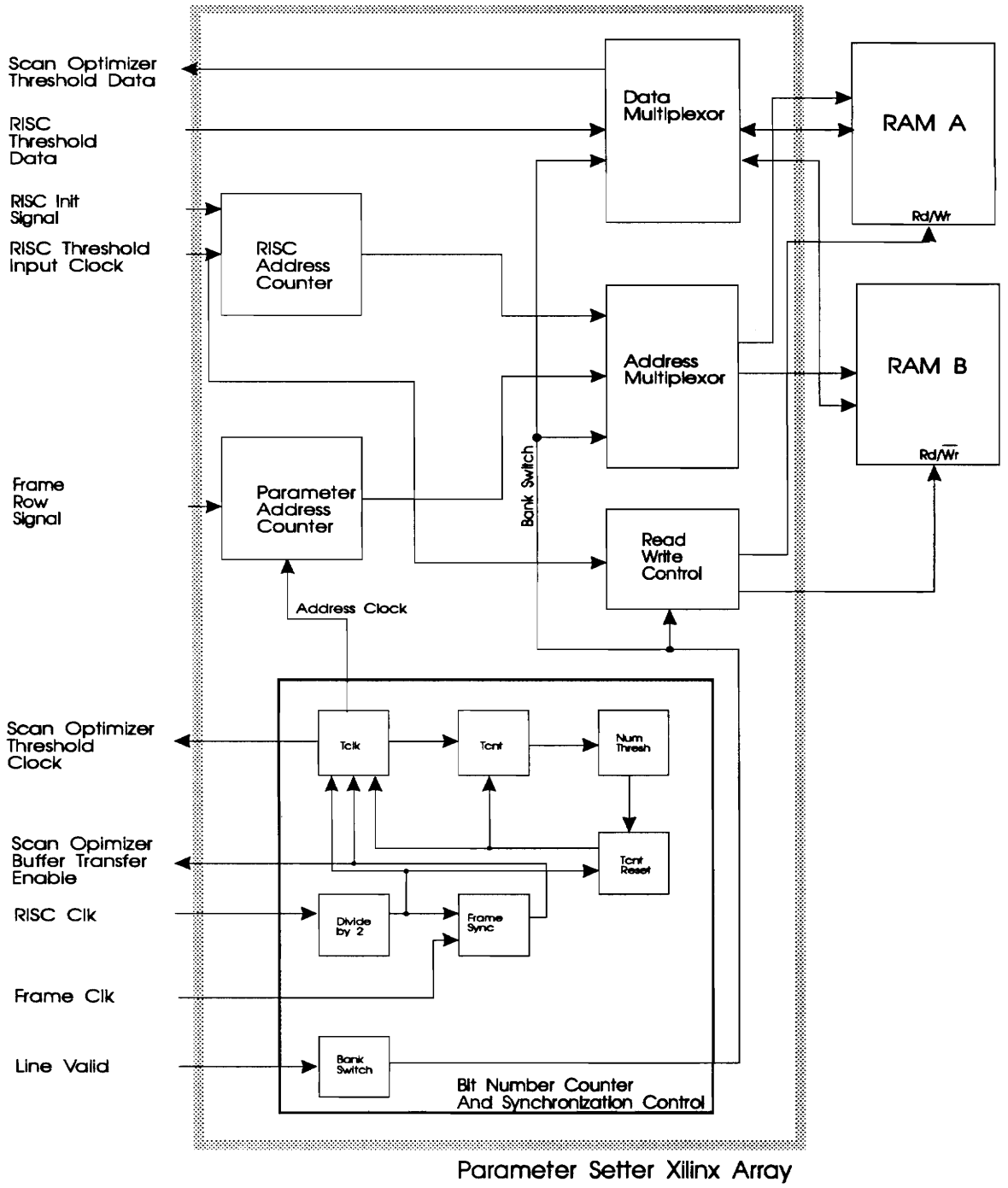


Figure 5.4. Parameter setter chip block diagram.

Data Flows:
Risc Threshold data/ Risc Threshold clock to counters to multiplexors to RAM
RISC 50 Mhz clock to Counter and Synchronization Control to Address counter to address multiplexor to RAM address to data multiplexor to Scan Optimizer
RISC 50 Mhz clock to counter and synchronization control to Scan Optimizer threshold clock and transfer enable

5.7.4.1. Two Buffers

Both of the tasks must be performed simultaneously. Both involve access to RAM, so two separate sections of RAM are required. One is for storing parameters from the RISC processor, and the other is for reading and setting the parameters in the Scan Optimizer. Due to simultaneity and no allowance for loss in data, two separate chips must be used.

To prevent the necessity of copying one RAM to the other when the tasks are finished, the task associated with the RAMs are switched. At the end of the last row in a sequence of row frames, the a set of multiplexors connected to the RAMs are switched. The RAM which was once being stored to is now being read and sent to the Scan Optimizer. The RAM which was being read in order to send parameters to the Scan Optimizer is now being overwritten by a new set of parameters from the RISC.

5.7.4.2. RISC Access

The RISC sends all parameters for a single row of image frames in one burst. To simplify the RISC instructions and board routing, only three signals are used for RISC parameter entry: one for data, one for clock, and one for reset. The reset is toggled at the beginning of the row parameter stream. Then the first frame's parameters are clocked in a bit at a time followed by subsequent frame's parameters.

The Parameter Setter accepts these parameters and stores them in the appropriate RAM device. A counter is used inside the parameter setter Xilinx array to keep track of the RAM address Some additional circuitry also exists in the Xilinx array in order to control the rest of the RAM.

5.7.4.3. Setting RAM Access and Process

Access to the RAM for the parameter setting process is accomplished in five sections: the RAM address counter, parameter bit number counter, address multiplexing,

data multiplexing, and miscellaneous internal and RAM controls. The RAM address counter counts the address which points to the particular bit of the parameter stream for the particular frame being sent. The parameter bit number counter counts the number of bits being sent in order to stop the counting process at the end of the parameter stream for each frame.

A 14-bit counter controls the address value of the RAM. It is reset at the end of the line and increments by one every time a bit of a parameter has been sent to the Scan Optimizer.

The parameter bit number counter, Tcnt0-5, counts which bit of a parameter set has been sent to the Scan Optimizer. There are 32 bits of parameters for the Scan Optimizer, so this value is compared to 32 in the NumThresh section. The TcntRst section then stops the counter, resets it, and waits for the next frame at which point it allows it to start counting again.

The miscellaneous controls are all created in the Xilinx array and consist of the clocks, the read/write signals, the parallel load signal, and the mode switch signal. The clocks for the two counters are identical. The parameter setting clock that is sent to the Scan Optimizer is derived from this clock and is delayed two cycles due to RAM delays. The RAM read and write pulses are also derived from the counter clocks. The parallel load signal for the Scan Optimizer is generated from the Frame Clock section and happens every 64 pixels. A simple flip-flop tied to the seventh bit of a line counter located elsewhere on the board is used for toggling a signal to indicate the RAM purposes; this changes every 64 lines.

5.7.4.4. Synchronization

Parameters are transmitted with a different clock than the scanner clock. This is done for two reasons. First of all, some scanners do not have clocks that operate between

lines. Secondly, a different clock will offer a better time division. With a 70 nanosecond clock, two cycles must be used to set each bit in the Scan Optimizer; for 30 parameter bits, this means 60 cycles are required. Since the frame size is 64 cycles, this does not allow much time for the parallel load or for the addition of new parameters. Therefore, the RISC clock is used for parameter sending. It is a 50 nanosecond period clock and is divided in half.

Careful synchronization must be done in the Xilinx array to accommodate this dual clock situation. The two sections it affects the most are the ThreshClk and TcntRst sections with the Frame Clock signal use. The Frame clock which is derived from the scanner clock is synchronized with the RISC clock into a single quick pulse in the ColFrame Sync part. The TcntRst uses this in order to send a signal after 32 pulses to the ThreshClk section to start the Tcnt0-5 counter again.

6. Results and Future Enhancements

6.1. Results

Chapter 2 states the requirements of the system. These concern operator intervention, adaptability, interfacing, speed, image results, and cost. The first three objectives have been satisfied easily by the method of implementation. The solution by the original algorithm to the problem of operator intervention was not compromised during implementation. Adaptability was kept by using Xilinx chips and a RISC processor for most of the design. Lastly, the Scan Optimizer interface specification was used as a standard for scanner interfacing so that the interfacing requirement would be met. The remaining three design objectives require more explanation.

6.1.1. Speed Requirement

The system was designed to operate at 15 megahertz, although the APS system has only been tested with the Fujitsu 3095 scanner which operates with a 5 megapixel output rate. Testing at a slower rate allowed the discrete logic timing and high speed RISC code to be not quite as critical. To operate at speeds up to 15 megahertz, some fine-tuning might be needed. For faster operation, such as with new 20 megahertz scanners, no fundamental design changes will be necessary, but newer, faster chips may need to be used for the RAMs and the processor.

6.1.2. Image Results

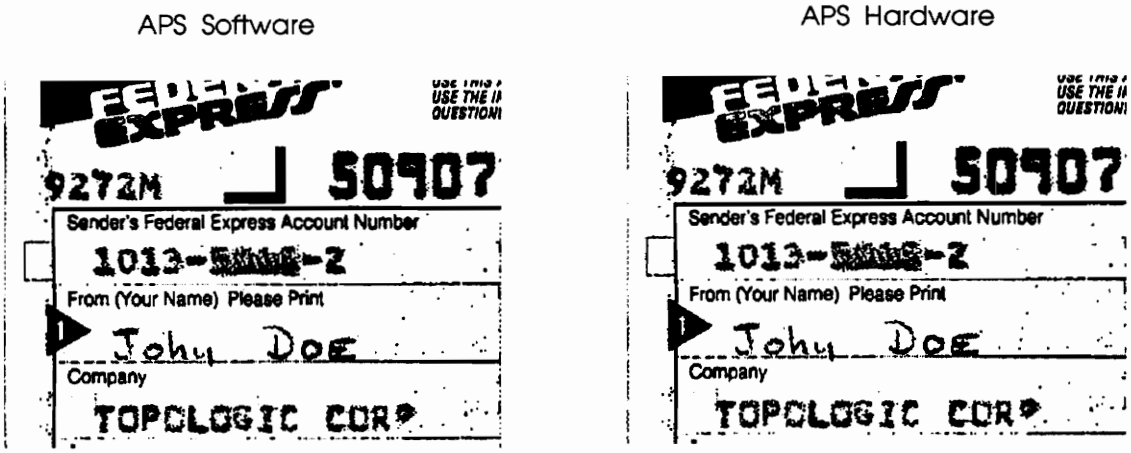
Images results were comparable with expectations. The Pseudo-Laplacian parameters of the Scan Optimizer were found accurately, and the output was similar to manually tuned outputs. Fine details were produced without excessive noise. The hardware with its approximations produced images very similar to the software results as shown in Figure 6.1.

One unexpected problem did occur, however, due to the original algorithm never being extensively tested for the blackfill parameter calculation. The Pseudo-Laplacian algorithm is an edge detection algorithm and produces outlines. These outlines need to be filled, and in the Scan Optimizer, this is the blackfill operation. It is performed by using a rough threshold of intensity values from the original image. The choice of this threshold is difficult and can be critical at times. In some documents, the threshold level was chosen by the APS hardware slightly lower or higher than optimum. This can be seen in the dark areas of Figure 6.1 (b).

The short-term solution is to set the blackfill level manually and is controlled on the handheld controller. A more sophisticated histogram analysis may solve the problem, but more research is needed. IPT is currently working on this problem.

6.1.3. Cost

Appendix B shows a bill of materials for the APS system valid in January 1991. The total cost for the components at that time was $1,394.31. In addition, there was a $2,075 nonrefundable cost for the first set of boards; this includes the initial printed circuit board and test fixture costs. This cost falls well below the maximum cost requirement of $3,000 per board, and can be reduced further if produced in large quantities.



(a) Federal Express Test Image Results

(b) Document Mix Test Image Results

Figure 6.1. APS software versus hardware images.

6.2. Future Enhancements

Hardware and software designs are never finished; improvements can always be made. Sometimes these involve design refinements, new technology, or just new ideas.

6.2.1. Fine-tuned Histogram Analysis

The original APS algorithm was tested with only a small number of images. This was due to the processing time required -- typically 15 to 30 minutes per image. The current method works well with the documents that have been tested, but it is possible that new documents may expose new problems. Now that the APS hardware has been built, it will be possible to test the algorithm rapidly with a large set of documents. This will permit fine-tuning of the process.

6.2.2. Dynamic Thresholding on the Scan Optimizer

To alleviate the critical threshold level calculation for the blackfill, dynamic thresholding is being implemented in the Scan Optimizer instead. This will make the blackfill operation robust, reducing the critical nature of this parameter, and thereby improving system operation. After the dynamic thresholding has been implemented in the Scan Optimizer, the APS formulas will be modified and the new APS system will be retested.

6.2.3. Simplification

Reducing cost and space is always a concern. The current design is of reasonable cost and size for initial production in small quantities. If this system is to be mass produced, cost and space will need to be reduced more. Some of this could come from

modifying the algorithm more, but the most reduction would come from using newer technology. VLSI, larger memories, faster RISC processor, and surface mount technology can drastically change the physical product while retaining the same functionality.

7. Conclusions

This thesis describes the implementation of the Automatic Parameter Setting algorithm for the IPT Scan Optimizer. This new system will improve the speed and performance of today's scanners, and will therefore improve the document archives for the future.

The resulting hardware is composed of three major sections: histogram generation, histogram processing, and interfacing. Dedicated digital design was used for histogram generation and parameter setting (a portion of the interfacing section), a RISC processor was used for histogram processing and parameter calculation, and interleaved dynamic RAM was used for a video input delay (another portion of the interfacing section). This hardware accomplishes the algorithm specification in real-time for scanners with pixel rates up to 15 megahertz.

The hardware has been built on PC-AT cards. To fit all the chips, two separate boards were built which mount together. These boards then mount to the Scan Optimizer board and the Scan Optimizer Interface board. The assembled board set occupies two slots in the host computer. This system was tested successfully with a Fujitsu 3095 page scanner which operates at a pixel rate of 5 megahertz.

Bibliography

1. AT&T, *WE DSP32c Digital Signal Processor Information Manual*, 1990.

2. C. Cornejo and R. Lee, "Comparing IBM's Microchannel and Apple's NuBus", *Byte Extra Edition- Inside the IBM PCs*, pp 87-92, 1987.

3. Cypress Semiconductor, *BiCMOS CMOS Data Book*, 1991, pp 7.1 - 7.45.

4. Fujistu, *MB86901 SPARC processor data sheet*, 1990.

5. Fujistu, *M3093E Image Scanner OEM Manual*, 1991.

6. L. B. Glass, "Inside EISA", *Byte*, pp. 417-425, November 1989.

7. R. Gonzalez and P. Wintz, *Digital Image Processing*, Massachusetts: Addison Wesley, 1987, pp. 331-359.

8. Hitachi, *IC Memory Data Book*, 1991.

9. Intel, *i486 Microprocessor*, 1989.

10. Image Processing Technologies, *IPT Scan Optimizer Ver 6.0*, 1991.

11. Image Processing Technologies, *IPT Scan Optimizer Internal Documentation*, 1991.

12. Image Processing Technologies, M. Nadler, N. Asimopoulos, and B. Cruikshank, "Method and Apparatus for Digitizing an Image": Patent, 1988.

13. P. Letellier, M. Nadler, and J. Abramatic, "The Telesign Project", *Proceedings of the IEEE*, April, 1985.

14. LSI Logic, *Digital Signal Processing (DSP) Data Book*, 1991, pp. 140-161.

15. T. Marshall and J. Potter, "How the Macintosh II NuBus Works", *Byte*, pp. mac38-mac51, December 1988.

16. National Archives and Record Administration (Research and Archive Evaluation Staff), *Optical Digital Image Storage System - Project Report*, March 1991.

17. R. Offen, *VLSI Image Processing*, New York: McGraw Hill, 1985, pp. 99-127.

18. J. Reynolds, "Building Tomorrow's Disk Controller Today", *Electronic Products*, December 15, 1987.

19. Ricoh Co., Ltd., *Image Scanner IS-400 Technical Manual*, 1986.

20. Ricoh Co., Ltd., *Image Scanner IS-400 Operation Manual Version 1.1*, 1986.

21. D. Smith, "User-Programmable Chips Take on a Broader Range of Applications", *VLSI Systems Design*, July, 1988.

22. P. Stucki, "Image Processing for Document Reproduction", *Advances in Digital Image Processing*, Plenum, 1978, pp 177-218.

23. J. E. Tatem, "Automatic Image Analysis Methods for Use with Local Operators", Master's Thesis - Virginia Polytechnic Institute and State University, 1990.

24. D. Wang, and A. Vagnucci, "Gradient Inverse Weighted Smoothing Scheme and the Evaluation of Its Performance", *Computer Graphics and Image Processing*, pp 167-181, 1981.

25. Xilinx, *The Programmable Gate Array Data Book*, 1991.

8. Appendix A

8.1. The Xilinx Chip

The Xilinx chip is a programmable logic cell array; this architecture is adaptable to many different designs. For a XC3030, there are 84 I/O blocks (one for each pin), 100 Configurable Logic Blocks, and reconfigurable interconnects between all the blocks as shown in Figure 8.1. This configurability enables the design to change easily, which would be useful if the APS system is ever upgraded.

The Xilinx array is configured every time at power up. This means design changes or design faults can be fixed without the loss of a single chip or change in hardware, thus speeding development time and cost. An example of a logic block configuration is shown in Figure 8.2 (a) and an example of an I/O block configuration is shown in Figure 8.2 (b).

For their capabilities, Xilinx chips are fast. Within a configurable logic block any 5-input, single-output boolean function can be processed in approximately 10 nanoseconds, and as fast as 5 nanoseconds with newer devices. The slowest part of the Xilinx chips is the interconnects between the configurable logic blocks, but with the sophisticated automatic place and route software available from Xilinx this is not much of a problem. Very complicated operations can be performed within the Xilinx array easily in 70 nanoseconds.

8.2. The Xilinx Developement Package

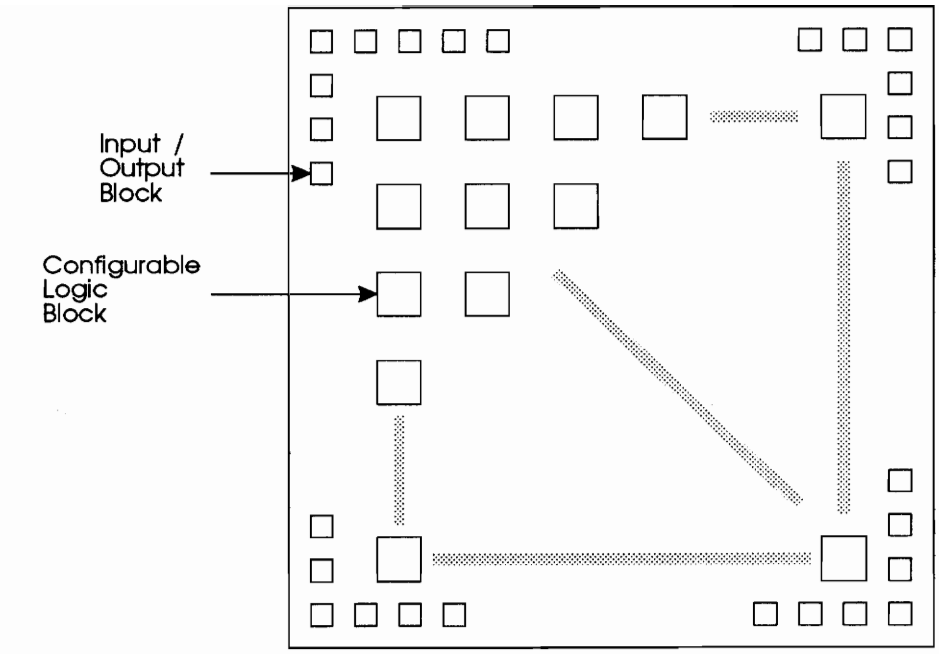
Xilinxes are programmed with the use of schematics and a translation package or directly with the XACT design package. For this project, extreme speeds were necessary, so the schematic method was not used. The XACT package presents the entire array of logic blocks, I/O blocks, and interconnects on the screen at once. From there each of these parts can be edited.

When a logic block is edited a new screen appears with all the options for the logic block. The logic blocks can be equated to minature PALs. They have two 4-input, 1-output function generators and there is an optional use of two flip-flops. The functions plus the configuration options can be edited here. Two examples of other options are the combination of the two function generators into one 5-input, 1-output function generator and the use of a multiplexor.

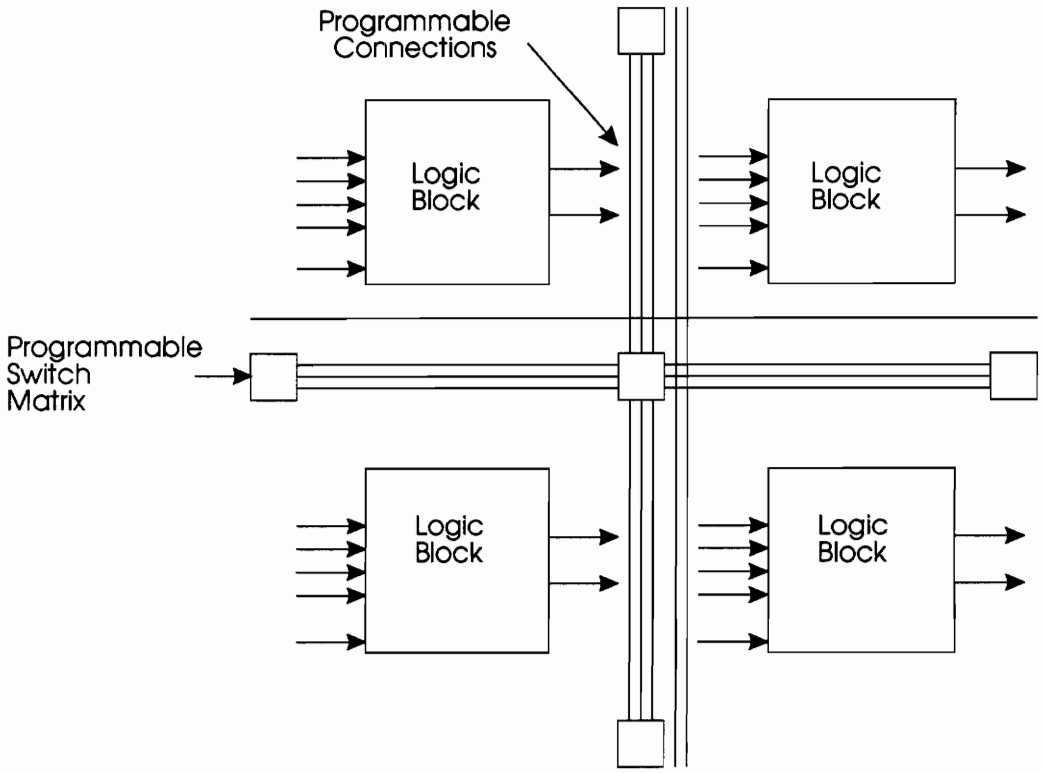
I/O blocks editing has a similar screen to the logic block editing. In this case, there are only configuration options; no functions. There is a choice of the use of input or output flip-flops and tri-statable input or output.

Interconnects can be routed manually, but this is very time consuming and autorouting is provided with the XACT package. Inside XACT, there is a simple autorouter that will route between two designated points. Outside of XACT there is an Automatic Place and Route program that will consider all routes at once and reroute the entire design; if asked it will also replace all the logic blocks. This system will route the most complex designs well.

After the design has been made, the MAKEBITS option of the XACT program is used. This makes the bit stream that is necessary for loading into the Xilinx at power up.

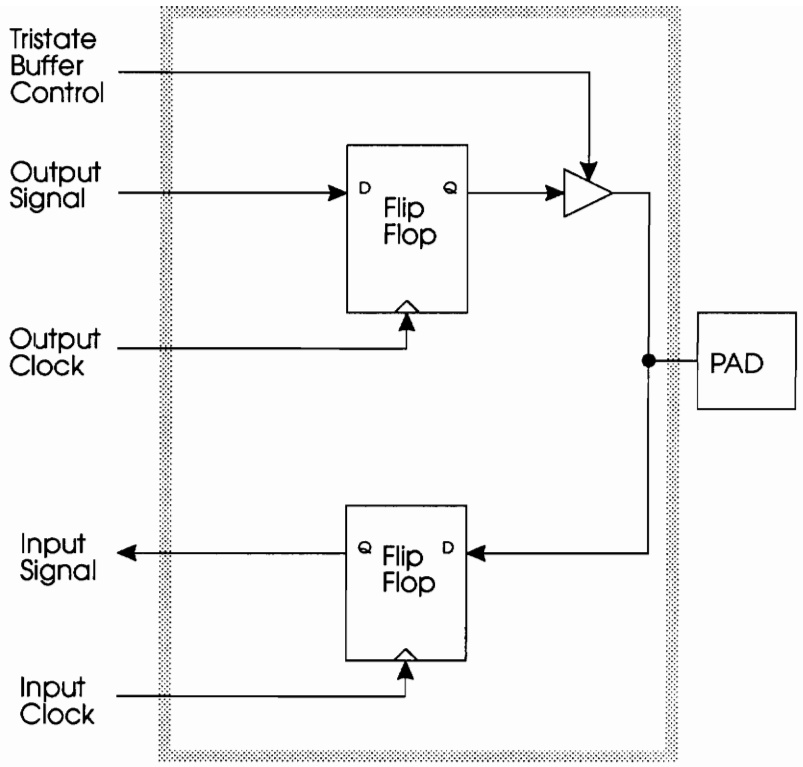


(a) Xilinx Logic Cell Array Architecture

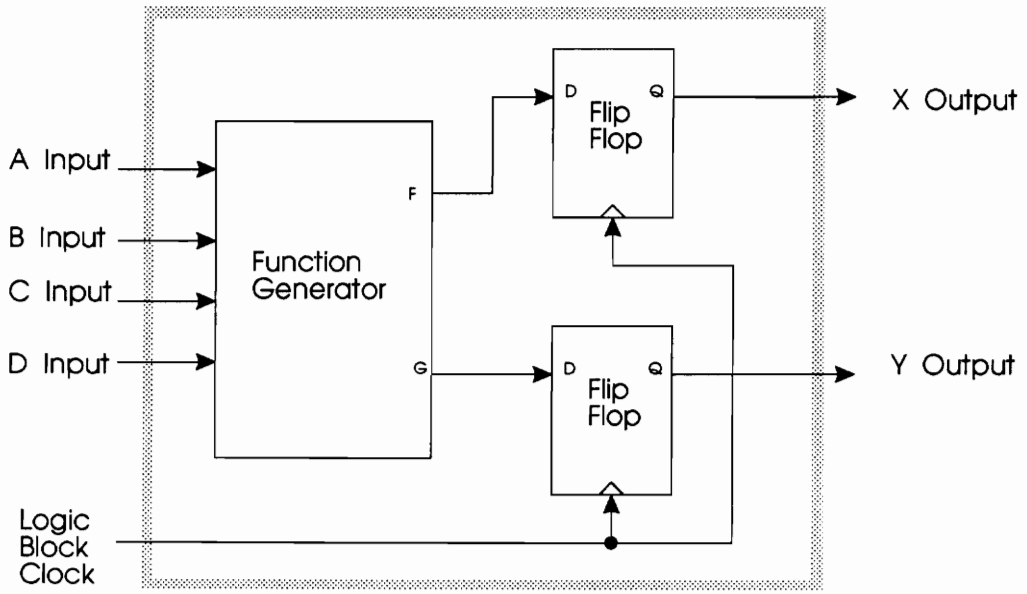


(b) Xilinx Interconnect Array Between Blocks

Figure 8.1. Xilinx logic cell array structure.



(a) Input / Output Block Example Configuration



(b) Logic Block Example Configuration

Figure 8.2. Xilinx internal block examples.

9. Appendix B

Bill of Materials for the APS board Set for January 1991

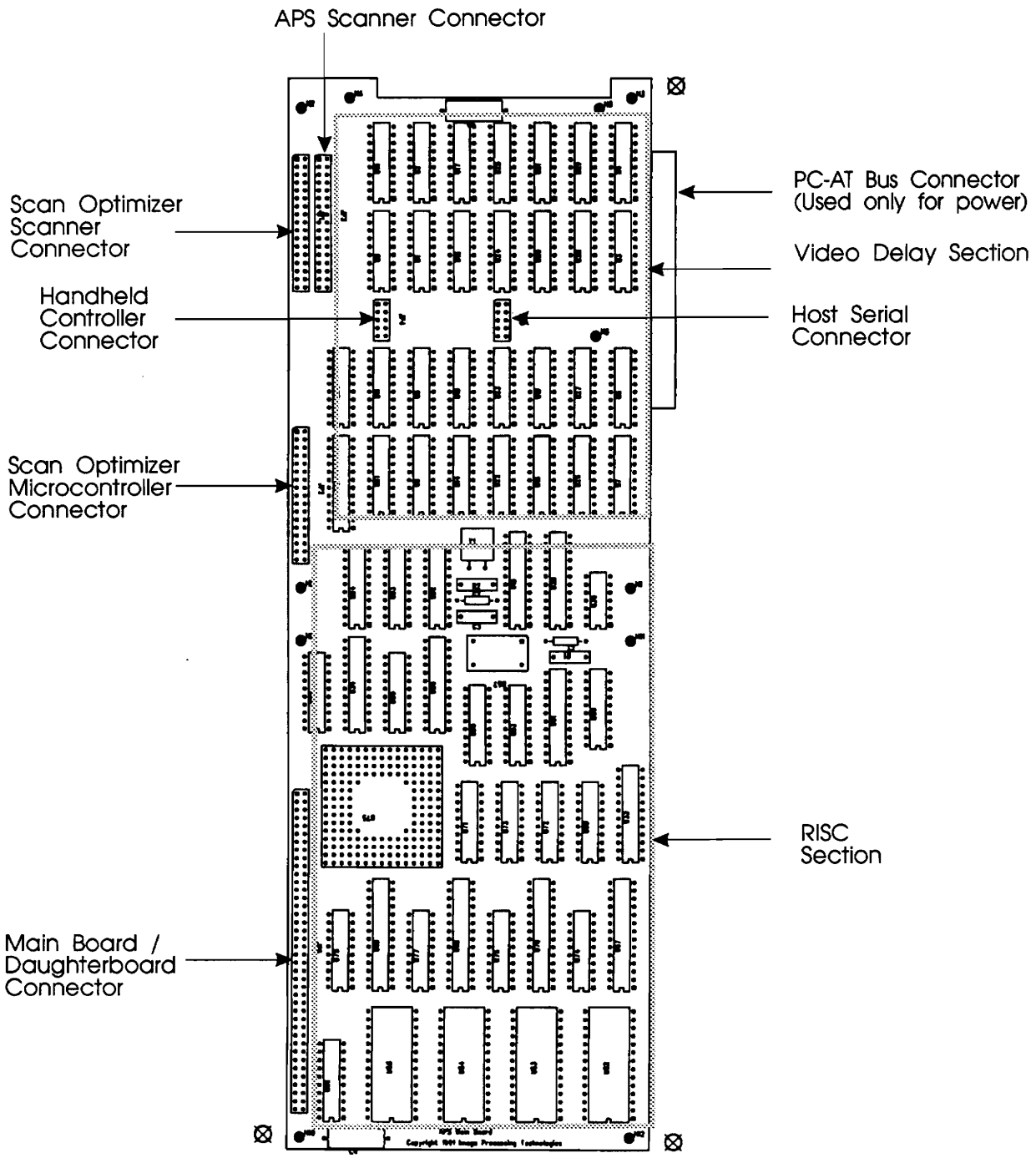
ITEM	DESCRIPTION	MAN. P/N	#	COST@	TOTAL COST
SRAM	64K x 4, w/OE_	CY7C196-25	6	13.98	83.88
XILINX	70 MHz	XC2018-70-PC84C	1	28.3	28.3
XILINX	100 MHz	XC3030-100-PC84 C	3	75	225
FLIP-FLOP	BICMOS	74BCT374	16	1.15	18.4
INVERT	HEX, DRIVER	74AS1004	1	1.2	1.2
INVERT	HEX	74AS04	1	0.56	0.56
BUFFER	BICMOS, HEX	74BCT244	8	1.2	9.6
LATCH		74F377	3	1.2	3.6
PAL		26V12	2	25	50
PAL		GAL16V8-15	5	3.75	18.75
PAL		GAL22V10-15	3	10.55	31.65
BUFFER	BICMOS	74BCT245	4	1.2	4.8
EPROM	UV ERASABLE, 32K x 8	27C256-120	4	5.45	21.8
SRAM	8K x 8	MT5C6408-20	6	8	48
UART	UART, 0.300" DIP	SCC2691-AC (SIGNETICS)	1	4.54	4.54
RS232	5 V, CMOS DRIVER/RECEIVER	MAX233CPP	1	4.75	4.75
SPARC	SPARC uPROCESSOR, 25 MHz	MB86901ACR-G25	1	136	136
COUNT	8 BIT	74F269	3	6	18
SRAM	16K x 1, 25 nS	CY7C167	2	13	26
DRAM	256K x 4, 80 nS, DIP	TMS44C256-80N	16	9.3	148.8
PCB#2	4 LAYER		1	125	125
PCB#1	4 LAYER		1	155	155
PWR PCB	AMP, UNIV. MATE-N-LOK, RT. ANGLE, 2 COND.	2-350942-0	1	1.29	1.29

PWR PLUG	AMP, UNIV. MATE-N-LOK, PLUG 2 COND.	350777-1	1	0.19	0.19
PWR PINS	SOCKET PINS	350550-2	2	0.41	0.82
HEADER	10 PIN, RT.ANGLE,	65625-110	2	0.96	1.92
HEADER RECPT.	DUPONT, DBL. ROW, 34 PIN RECPT.	68683-317	2	4.52	9.04
HEADER	DUPONT,DBL. ROW, 34 PIN	67997-134	1	1.48	1.48
HEADER RECPT.	DUPONT, DBL. ROW, 80 PIN RECPT.	68683-340	1	10.38	10.38
HEADER	DUPONT, DBL. ROW, 80 PIN	67997-140	1	2.21	2.21
SOCKET	CM, 28 PIN, 0.600", W/CAP	CS628-LP-TG.1Z	4	4.09	16.36
SOCKET	CM, 28 PIN, 0.300", W/CAP	CS328-LP-TG.1Z	14	3.55	49.7
SOCKET	CM, 20 PIN, 0.300", W/CAP	CS320-LP-TG.1Z	23	1.77	40.71
SOCKET	CM, 24 PIN, 0.300", W/CAP	CS324-LP-TG.1Z	4	3.09	12.36
SOCKET	CM, 200 PIN PGA, W/CAP	UDPGA-200-LG-GG -.111	1	25	25
CAP IC BYPASS	ROGERS, UNDER IC, 20 PIN	303A20	27	0.71	19.17
CAP IC BYPASS	ROGERS, UNDER IC, 84 PIN PLCC	253ADD	4	2.84	11.36
OSC.	CONNER 2 TTL LOADS, 14 PIN, 50 MHz	HC15R8-50M	1	25	25
CRYST.	4 MHz, H-18 CAN		1	0.77	0.77
CAP BD BYPASS	KEMET, TANTALUM MOLDED, AXIAL 6V, 100 uF	T322E107K006AS	4	0.73	2.92
		TOTAL			1394.31

NOTE: Additional Cost for first board set= $2075.00

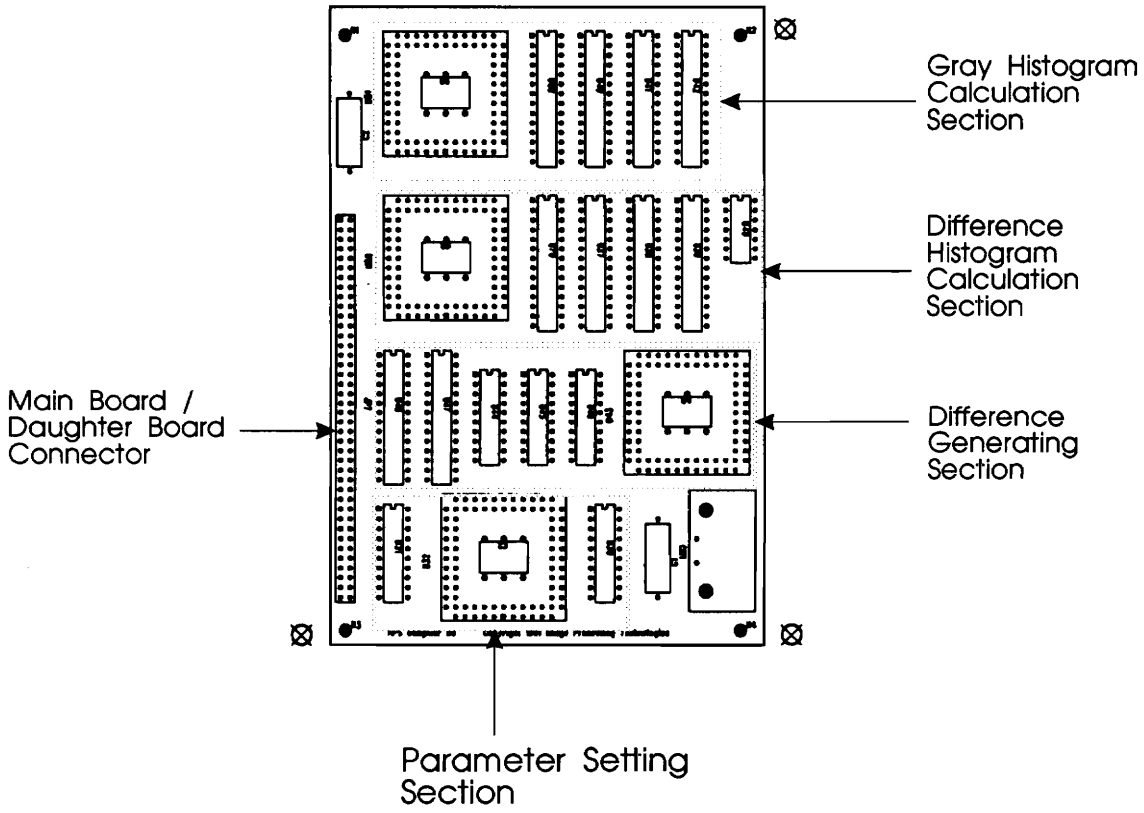
10. Appendix C

Following is a printed circuit board layout of the APS board set. To fit the components into a PC-AT bus structure it was necessary to make a two-board set. The main board contains the RISC processor with its support circuitry and the Video Delay. The daughter board contains the Histogram Calculation and Parameter Setting systems.



APS Main Board Printed Circuit Board

Half True Size Shown



Histogram Generation and Parameter Setting
Printed Circuit Board

Half True Size Shown

11. Appendix D

Following are the schematics for the APS board set. The first twelve pages are for the main board, and the following ten pages are for the daughterboard. They are formed into a hierarchical style where higher level schematics have sheet symbols which point to related lower level schematics.

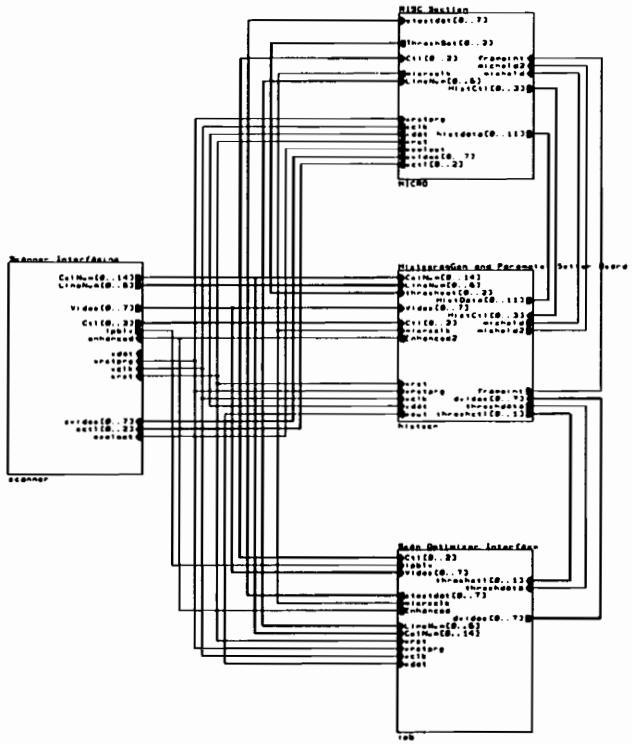
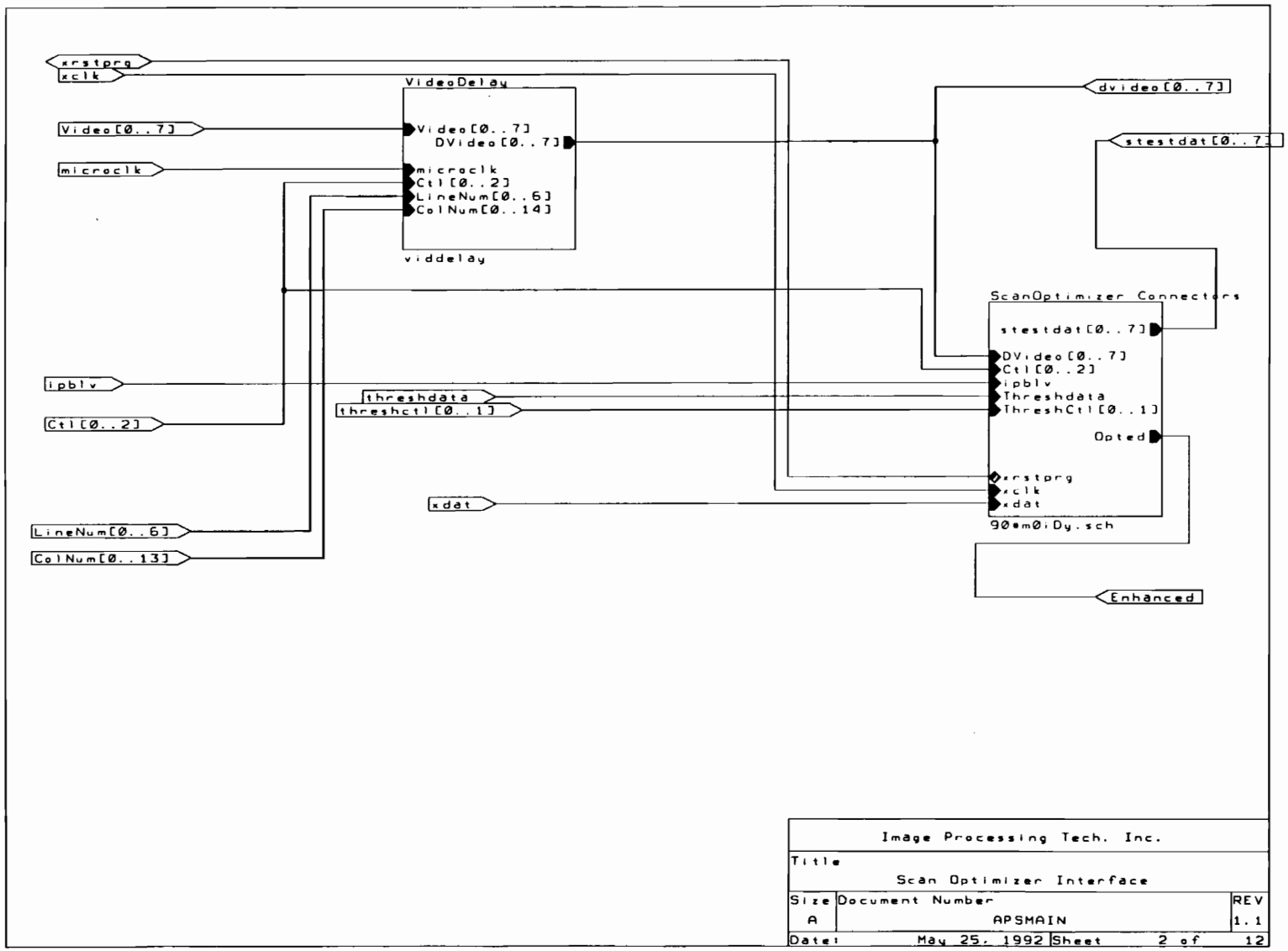
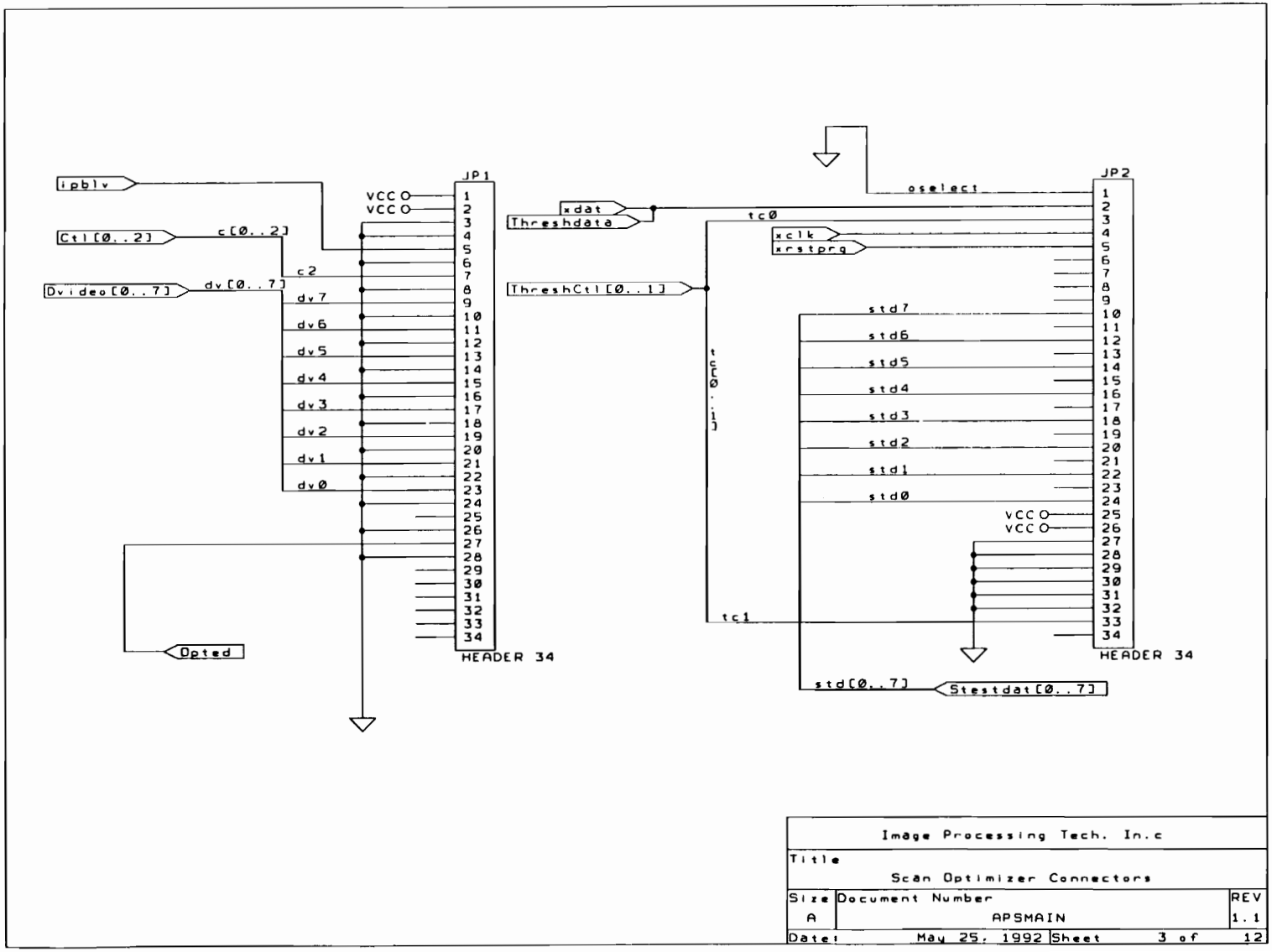
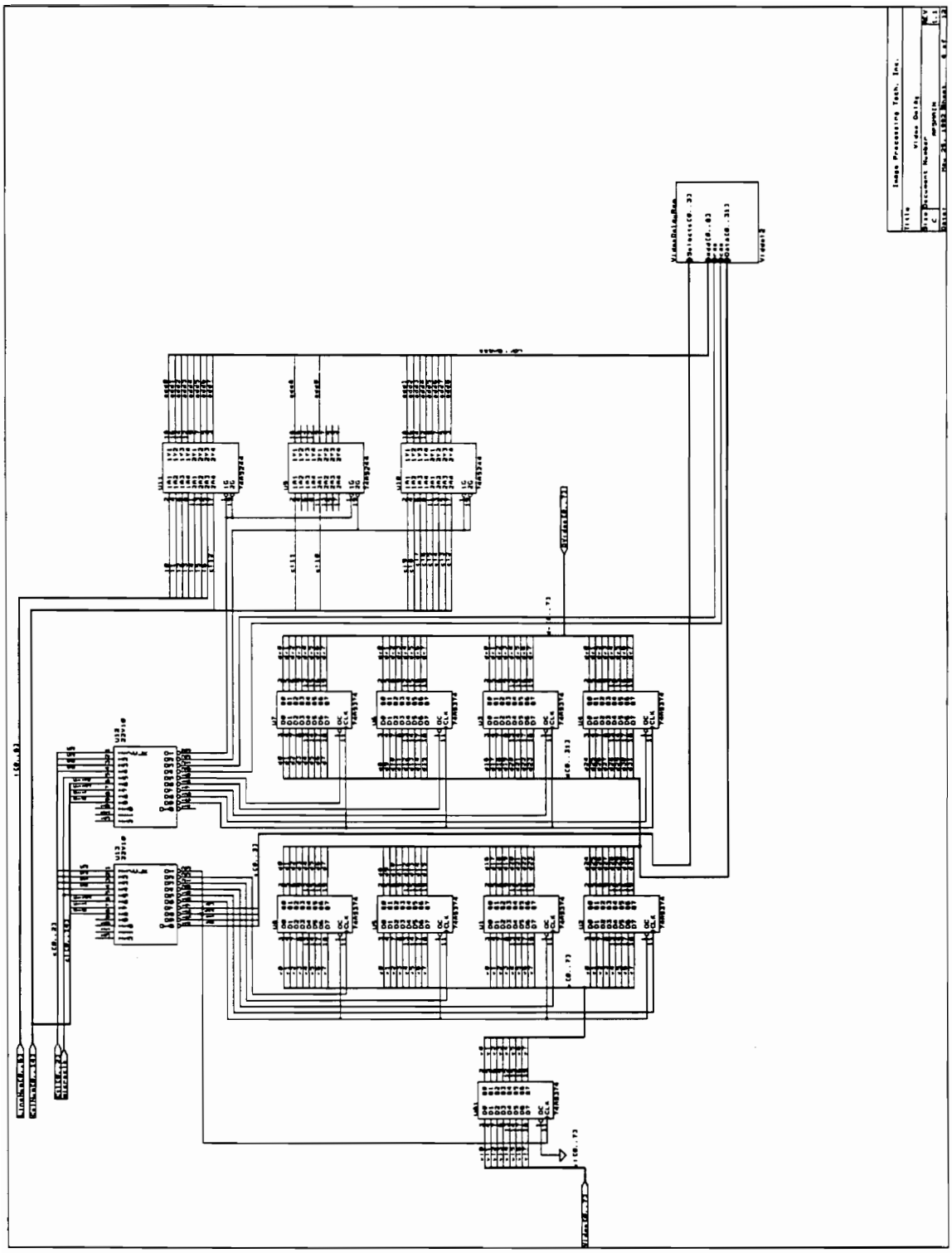


Image Processing Tech. Inc.		
Title: Automatic Parameter Setter Main Board		
Size: C	Document Number: APSMAIN	Rev: 5.1
Date: May 29, 1992	Sheet: 1 of 12	


xrstprg
xclk
VideoDelay
Video[0..7]
DVideo[0..7]
dvideo[0..7]
Video[0..7]
stestdat[0..7]
microclk
microclk
Ctl[0..2]
LineNum[0..6]
ColNum[0..14]
viddelay
ScanOptimizer Connectors
stestdat[0..7]
DVideo[0..7]
Ctl[0..2]
ipblv
Threshdata
ThreshCtl[0..1]
Opted
xrstprg
xclk
xdat
90sm0iDy.sch
ipblv
threshdata
threshctl[0..1]
Ctl[0..2]
xdat
LineNum[0..6]
ColNum[0..13]
Enhanced
Image Processing Tech. Inc.
Title
Scan Optimizer Interface
Size
A
Document Number
APSMAIN
REV
1.1
Date:
May 25, 1992
Sheet
2 of
12


ipblv
Ctl[0..2]
c[0..2]
c2
Dvideo[0..7]
dv[0..7]
dv7
dv6
dv5
dv4
dv3
dv2
dv1
dv0
Opted
JP1
VCC
VCC
HEADER 34
xdat
Threshdata
ThreshCtl[0..1]
tc0
tc[0..1]
tc1
oselect
xclk
xrstprg
std7
std6
std5
std4
std3
std2
std1
std0
std[0..7]
Stestdat[0..7]
JP2
HEADER 34
Image Processing Tech. Inc
Title
Scan Optimizer Connectors
Size
A
Document Number
APSMAIN
REV
1.1
Date: May 25, 1992
Sheet 3 of 12



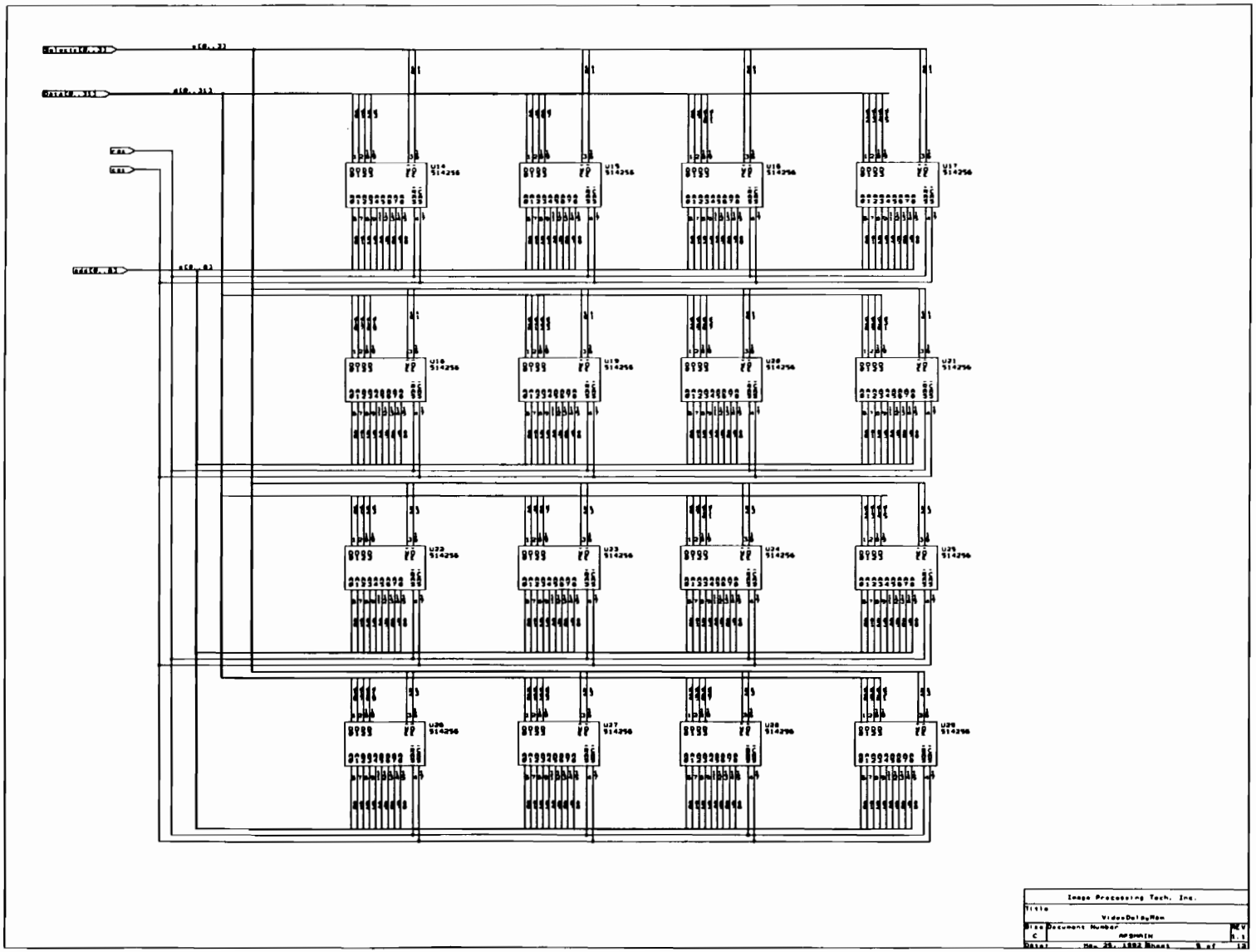


Image Processing Tech. Inc.
Title
VideoDelayMem
Size
C
Document Number
AP3MEM
REV
1.1
Date: Mon, 25, 1992 Sheet 9 of 12

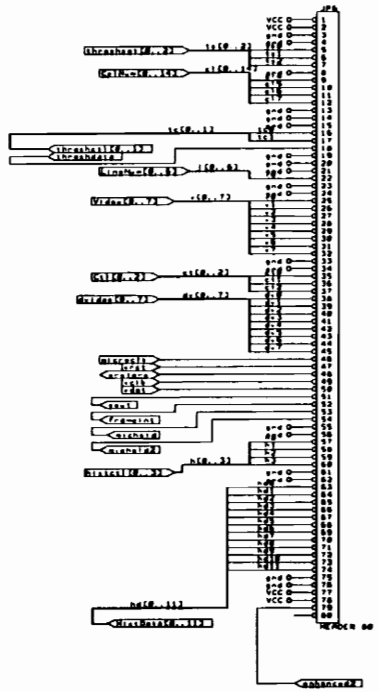


Image Processing Tech. Inc.		
Title: Histogramgen and Parameter Setter Board		
Size: C	Document Number: HPSMAIN	REV: B.1
Date: Mar. 29, 1988	Sheet 8 of 12	

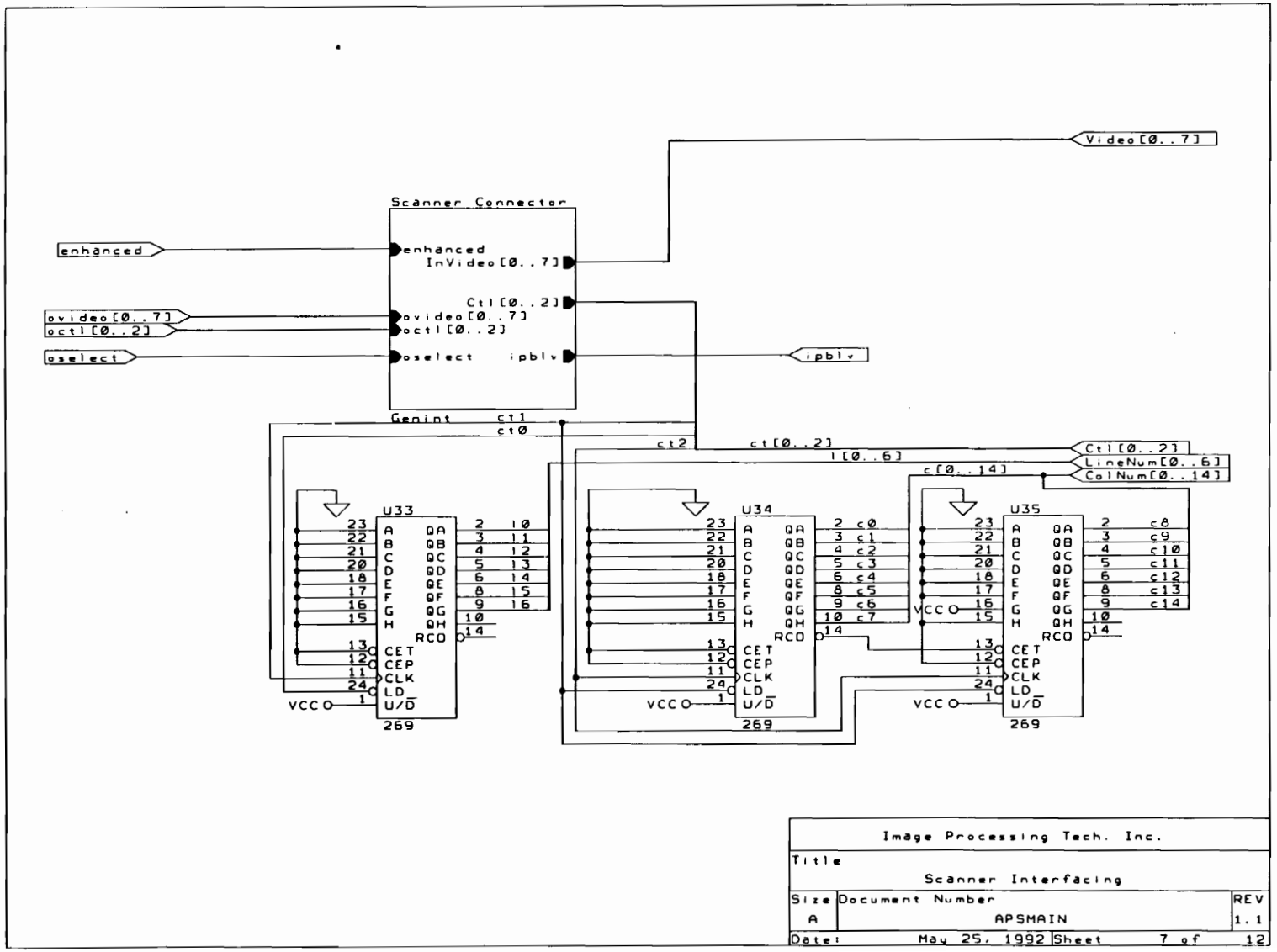
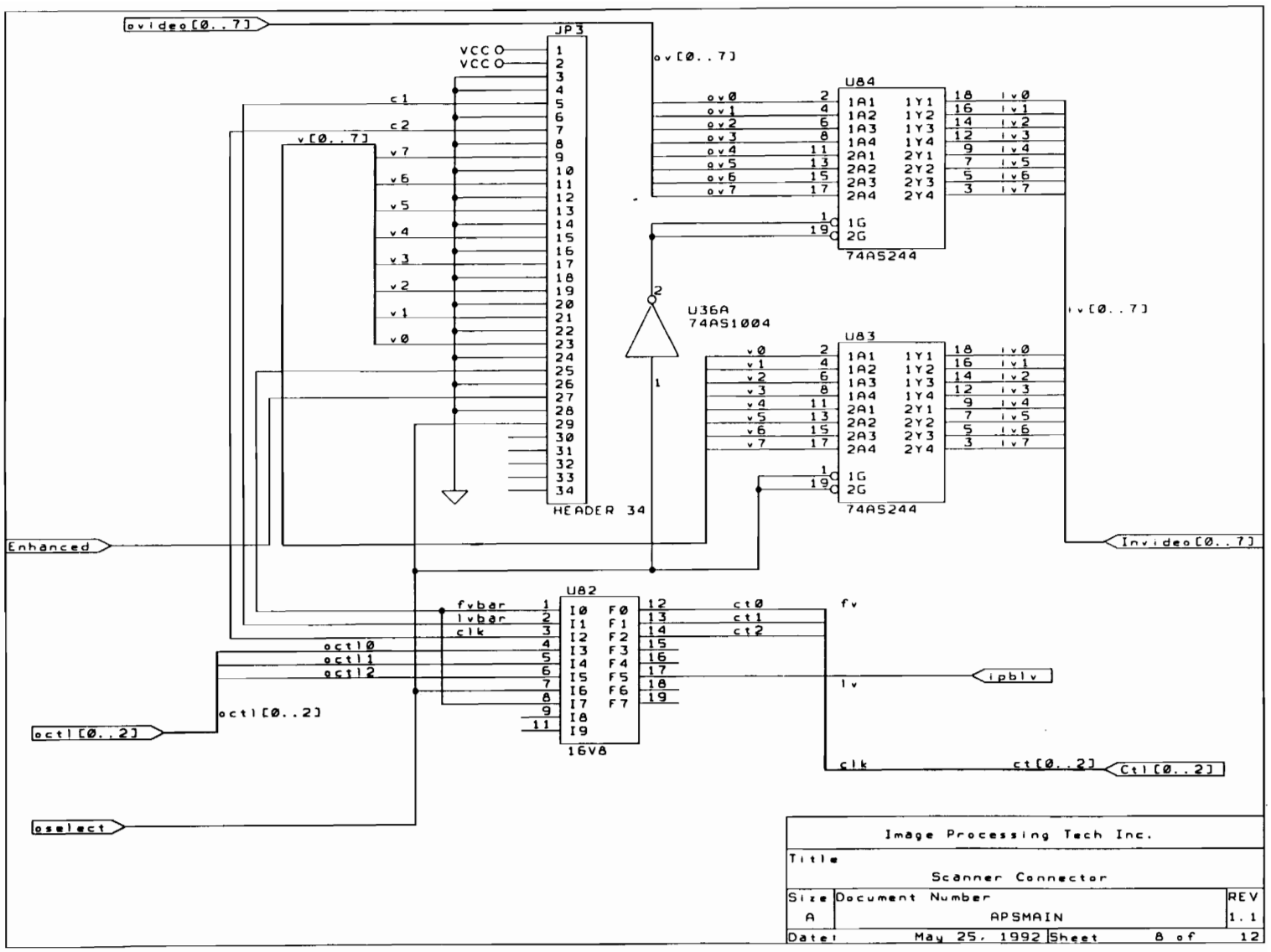
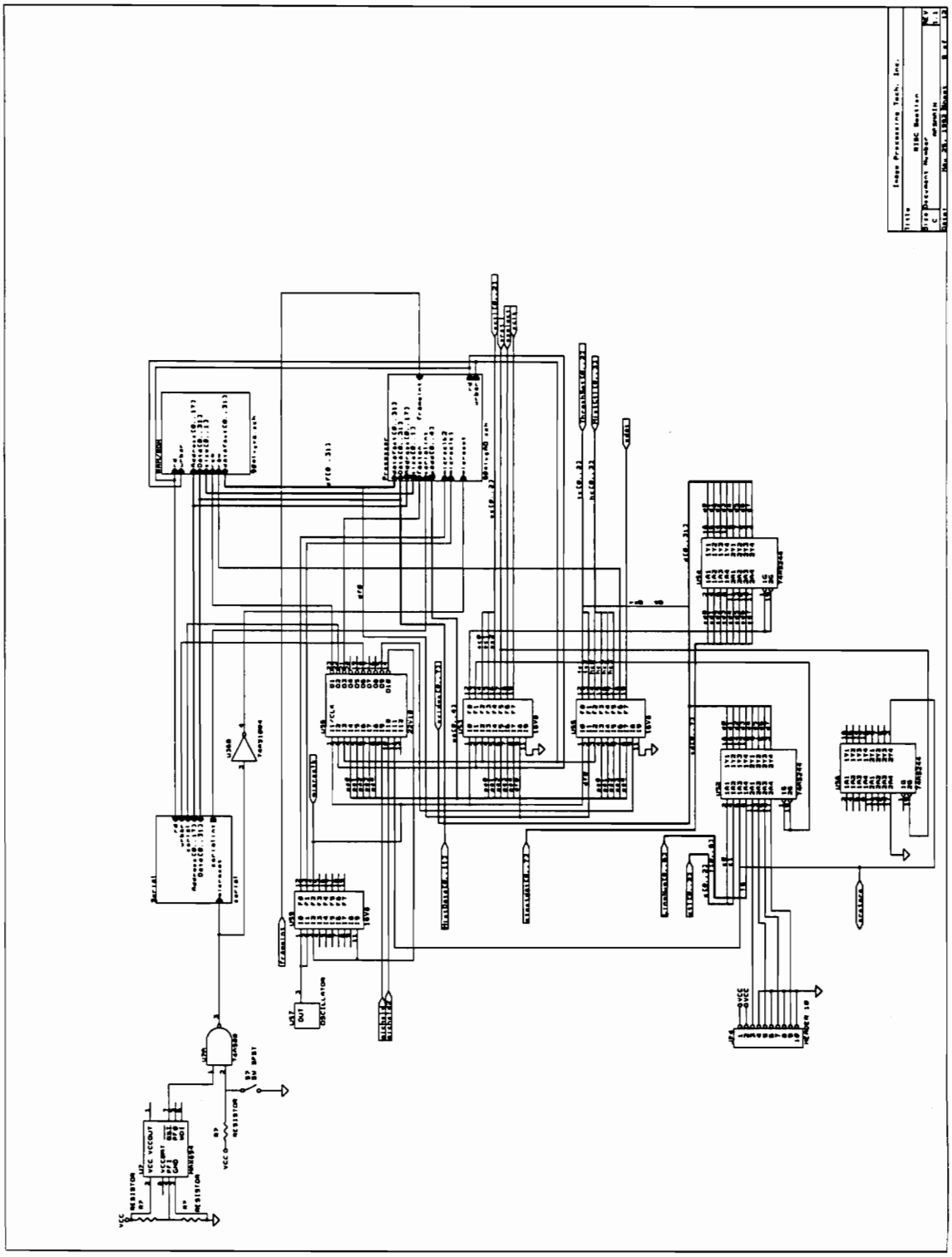
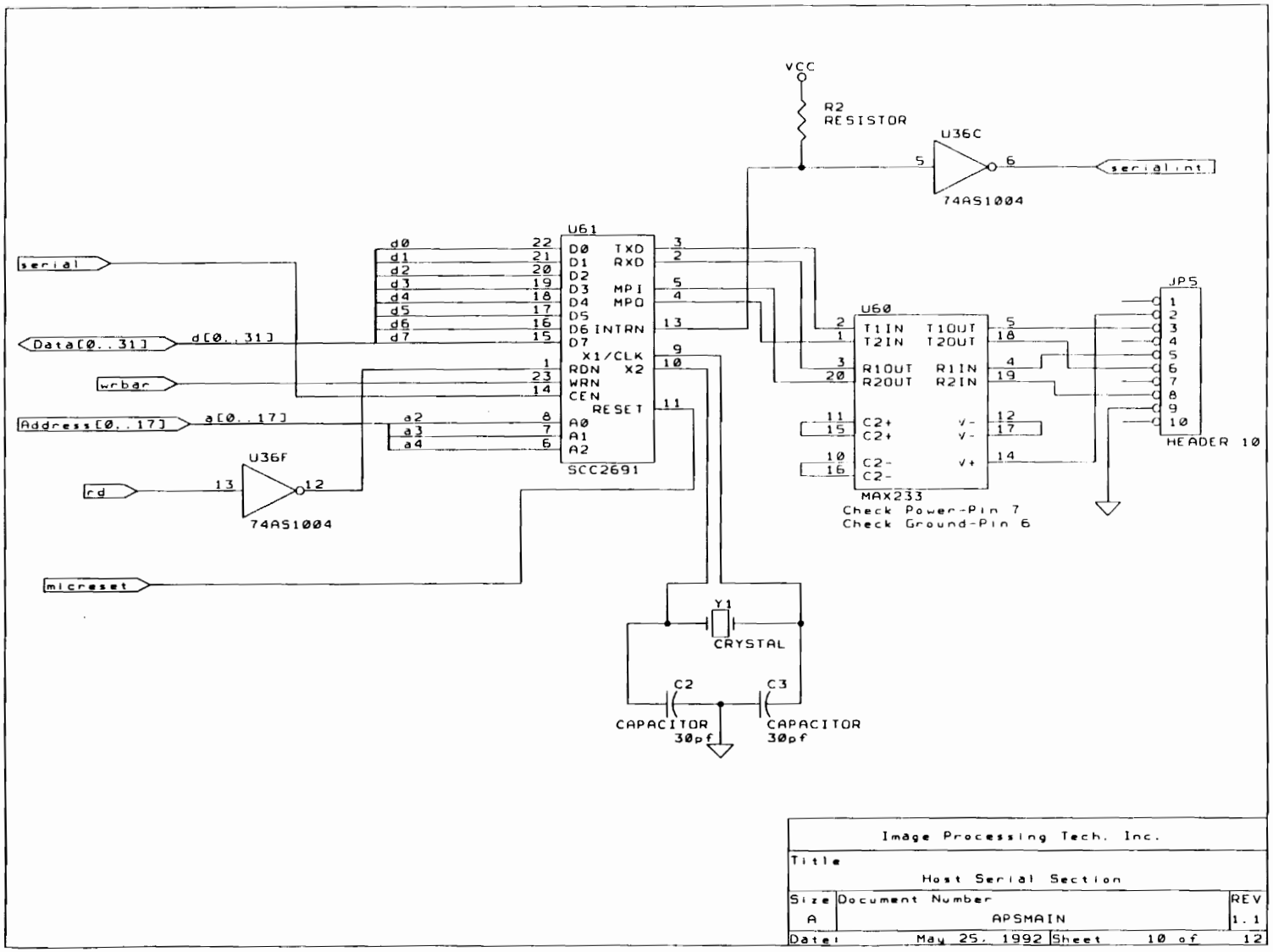


Image Processing Tech. Inc.		
Title: Scanner Interfacing		
Size: A	Document Number: APSMAIN	REV: 1.1
Date: May 25, 1992	Sheet 7 of 12	


ovideo[0..7]
JP3
VCC
VCC
HEADER 34
ov[0..7]
U84
74AS244
U36A
74AS1004
U83
74AS244
iv[0..7]
Invideo[0..7]
Enhanced
U82
16V8
fvbar
lvbar
clk
oct10
oct11
oct12
oct1[0..2]
oct1[0..2]
oselect
ct0
ct1
ct2
fv
lv
ipblv
clk
ct[0..2]
Ctl[0..2]
Image Processing Tech Inc.
Title
Scanner Connector
Size
A
Document Number
APSMAIN
REV
1.1
Date:
May 25, 1992
Sheet
8 of
12




VCC
R2
RESISTOR
U36C
74AS1004
serialint
U61
SCC2691
serial
Data[0..31]
d[0..31]
wrbar
Address[0..17]
a[0..17]
rd
U36F
74AS1004
micreset
U60
MAX233
Check Power-Pin 7
Check Ground-Pin 6
JP5
HEADER 10
Y1
CRYSTAL
C2
C3
CAPACITOR
30pf
CAPACITOR
30pf
Image Processing Tech. Inc.
Title
Host Serial Section
Size
A
Document Number
APSMAIN
REV
1.1
Date:
May 25, 1992
Sheet
10 of
12

U6

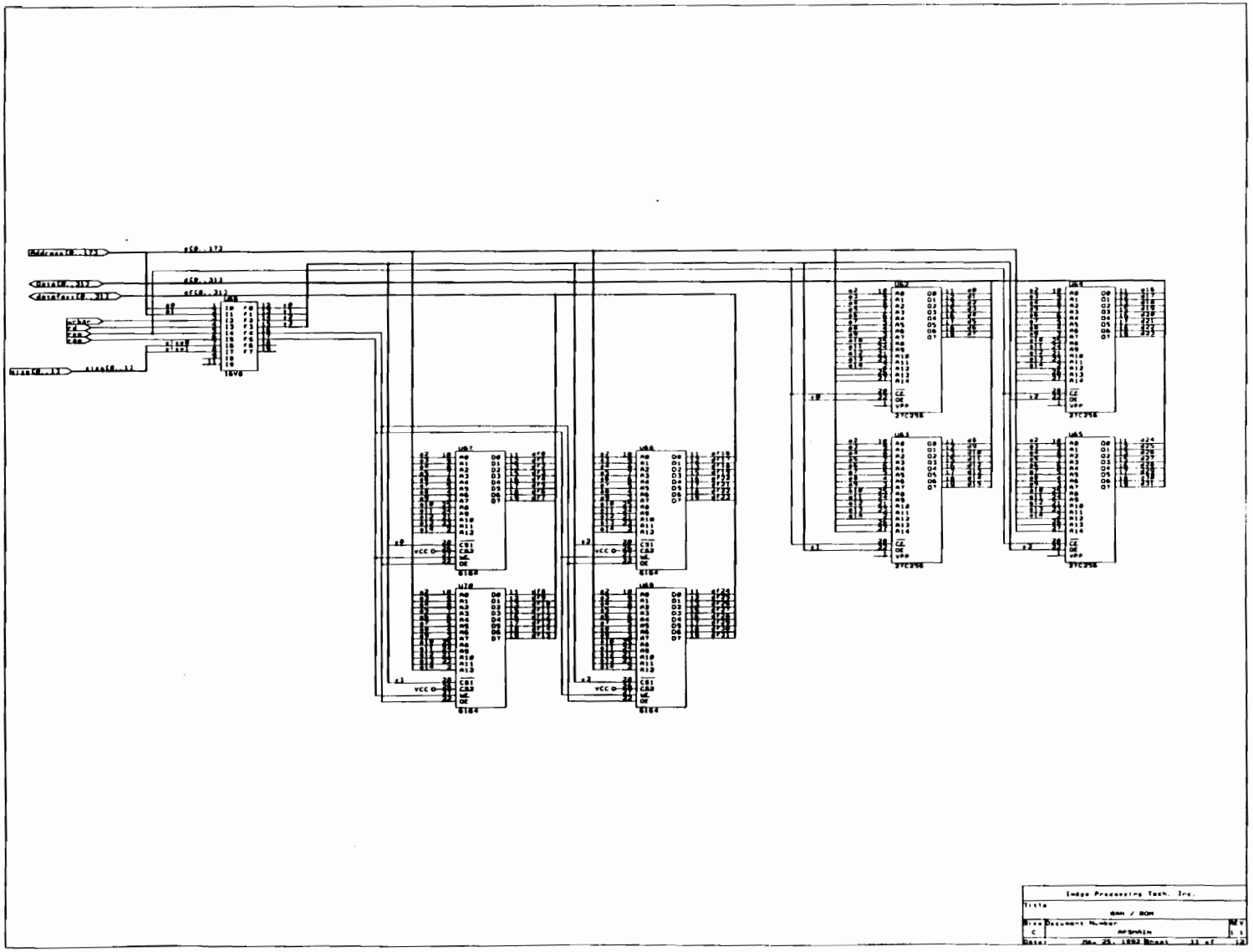
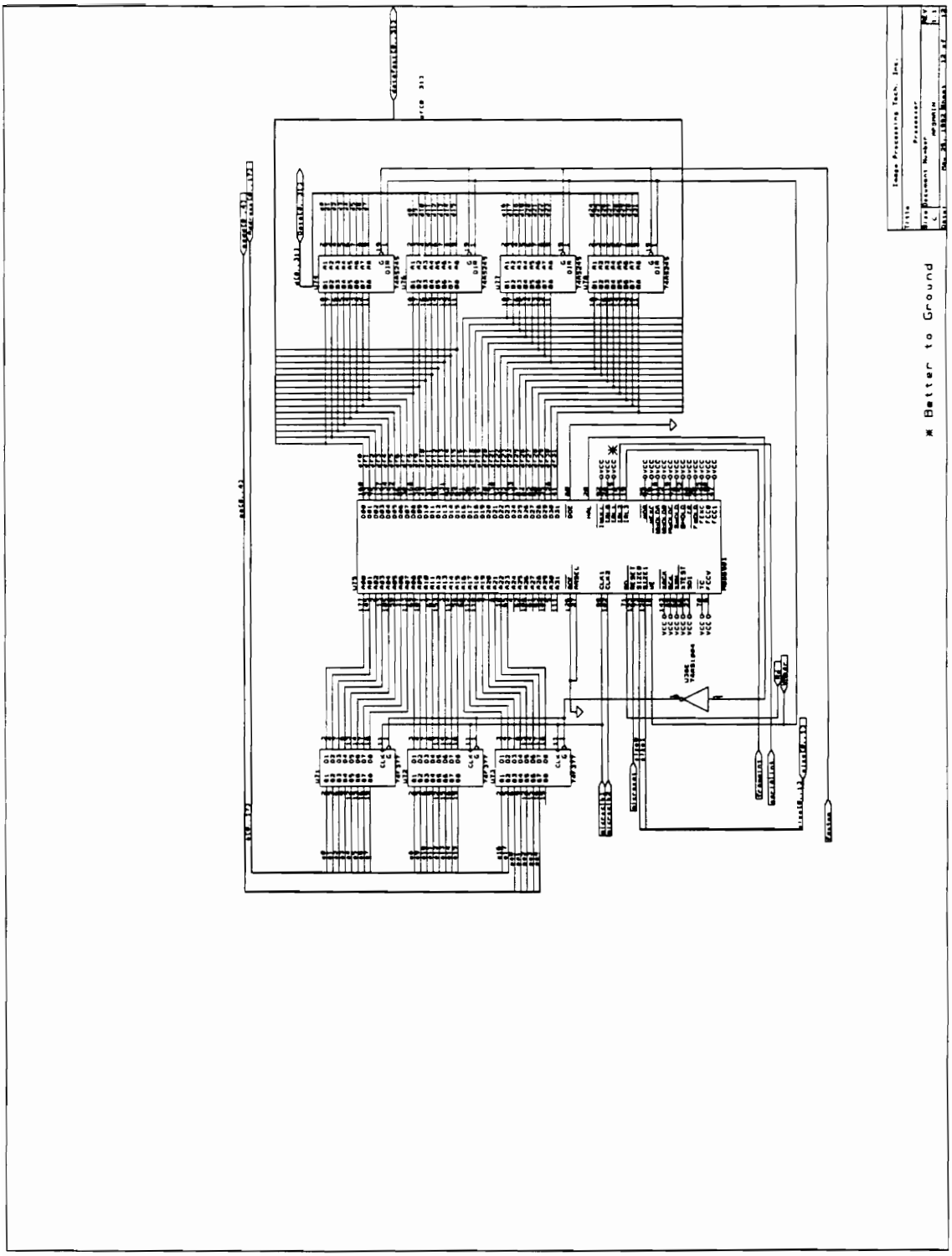
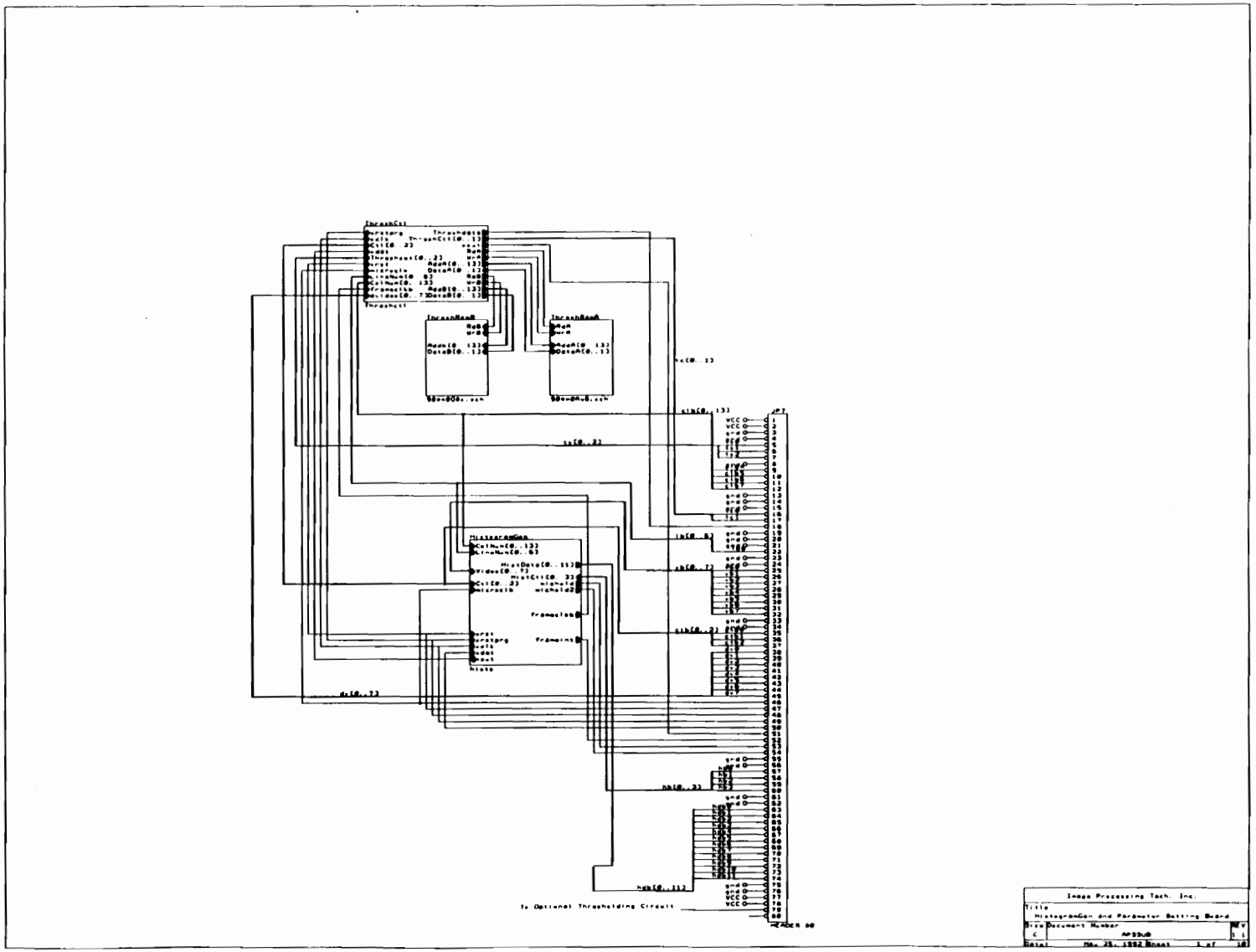


Image Processing Tech. Inc.
Title
RAM / ROM
Size
C
Document Number
MPSMAIN
Rev
1.1
Date: Mon 25, 1992 Sheet 11 of 12


* Better to Ground


IncashCct
IncashRam8
IncashRam8
HistogramGen
JP7
To Optional Thresholding Circuit
HEADER 80
Image Processing Tech. Inc.
Title
HistogramGen and Parameter Setting Board
Size
Document Number
Rev
Date: Mar 25, 1992 Sheet 1 of 10

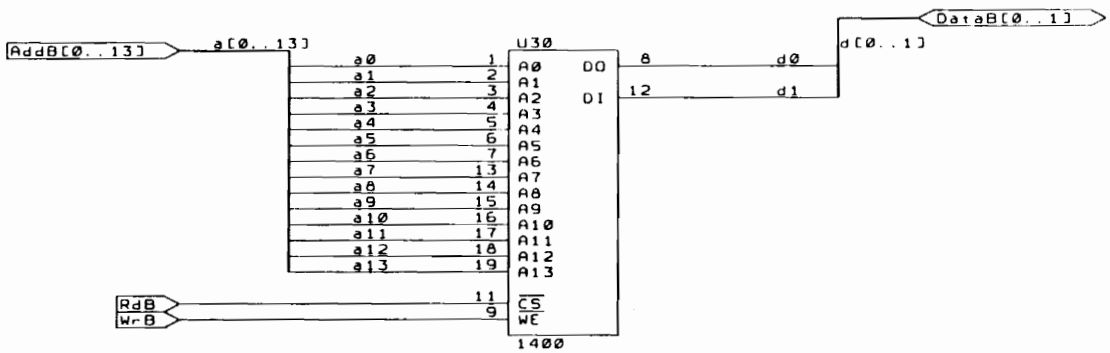


Image Processing Tech. Inc.		
Title: ThreshRamB		
Size: A	Document Number: APSSUB	REV: 1.1
Date: May 25, 1992	Sheet: 2 of 10	

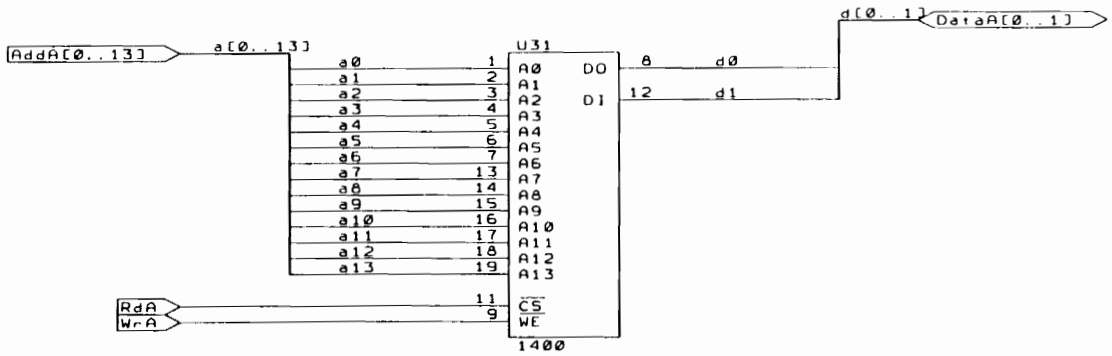
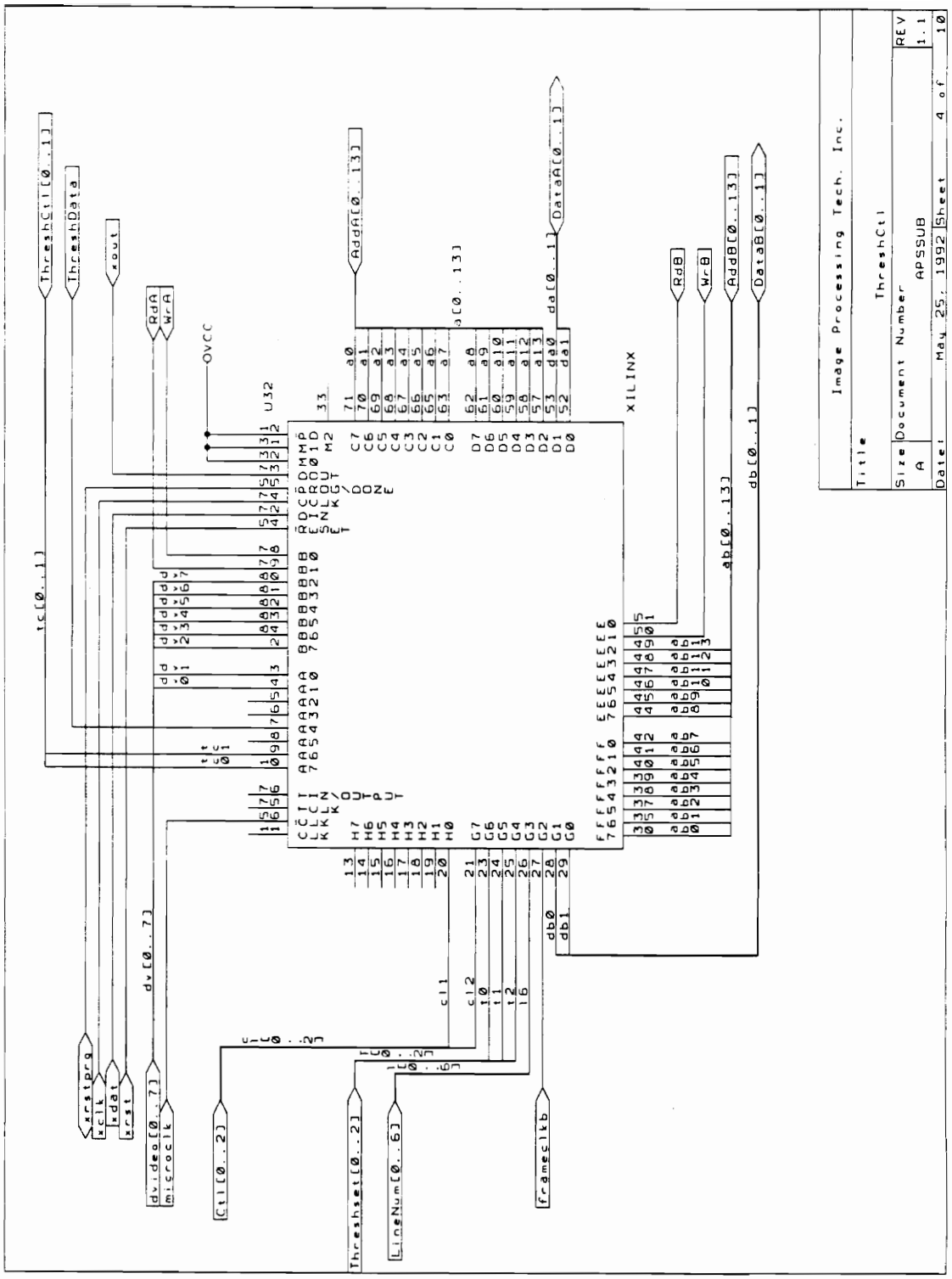
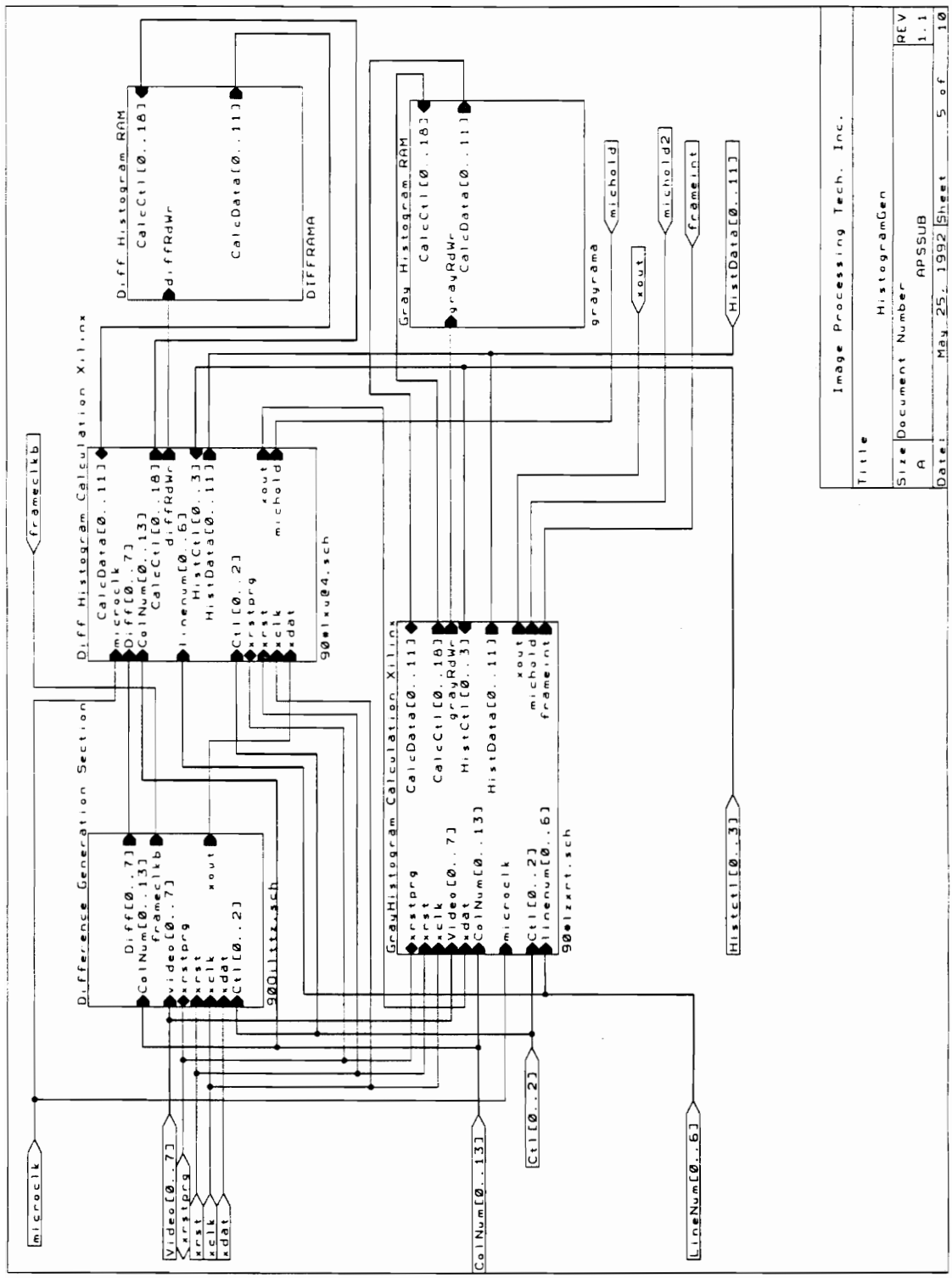


Image Processing Tech. Inc.		
Title: ThreshRamA		
Size: A	Document Number: APSSUB	REV: 1.1
Date: May 25, 1992	Sheet 3 of 10	


ThreshCtl[0..1]
ThreshData
xout
RdA
WrA
Vcc
tc[0..1]
U32
AddA[0..13]
DataA[0..1]
RdB
WrB
AddB[0..13]
DataB[0..1]
db[0..1]
ab[0..13]
XILINX
xrstprg
xclk
xdat
xrst
dv[0..7]
microclk
Ctl[0..2]
Threshset[0..2]
LineNum[0..6]
frameclkb
Image Processing Tech. Inc.
Title
ThreshCtl
Size
A
Document Number
APSSUB
REV
1.1
Date:
May 25, 1992
Sheet
4 of 10


Diff Histogram RAM
CalcCtl[0..18]
diffRdWr
CalcData[0..11]
DIFFRAMA
Diff Histogram Calculation Xilinx
CalcData[0..11]
microclk
Diff[0..7]
ColNum[0..13]
CalcCtl[0..18]
diffRdWr
LineNum[0..6]
HistCtl[0..3]
HistData[0..11]
Ctl[0..2]
xrstprg
xrst
xclk
xdat
xout
michold
900lxu04.sch
framecIkb
Difference Generation Section
Diff[0..7]
ColNum[0..13]
framecIkb
video[0..7]
xrstprg
xrst
xclk
xdat
xout
Ctl[0..2]
900lttr.sch
GrayHistogram Calculation Xilinx
xrstprg
xrst
xclk
Video[0..7]
xdat
ColNum[0..13]
microclk
CalcData[0..11]
CalcCtl[0..18]
grayRdWr
HistCtl[0..3]
HistData[0..11]
xout
michold
framecnt
Ctl[0..2]
LineNum[0..6]
900lzxrt.sch
Gray Histogram RAM
CalcCtl[0..18]
grayRdWr
CalcData[0..11]
grayrama
michold
xout
michold2
framecnt
HistData[0..11]
Histctl[0..3]
microclk
Video[0..7]
xrstprg
xrst
xclk
xdat
ColNum[0..13]
Ctl[0..2]
LineNum[0..6]
Image Processing Tech. Inc.
Title
HistogramGen
Size
A
Document Number
APSSUB
REV
1.1
Date: May 25, 1992 Sheet 5 of 10

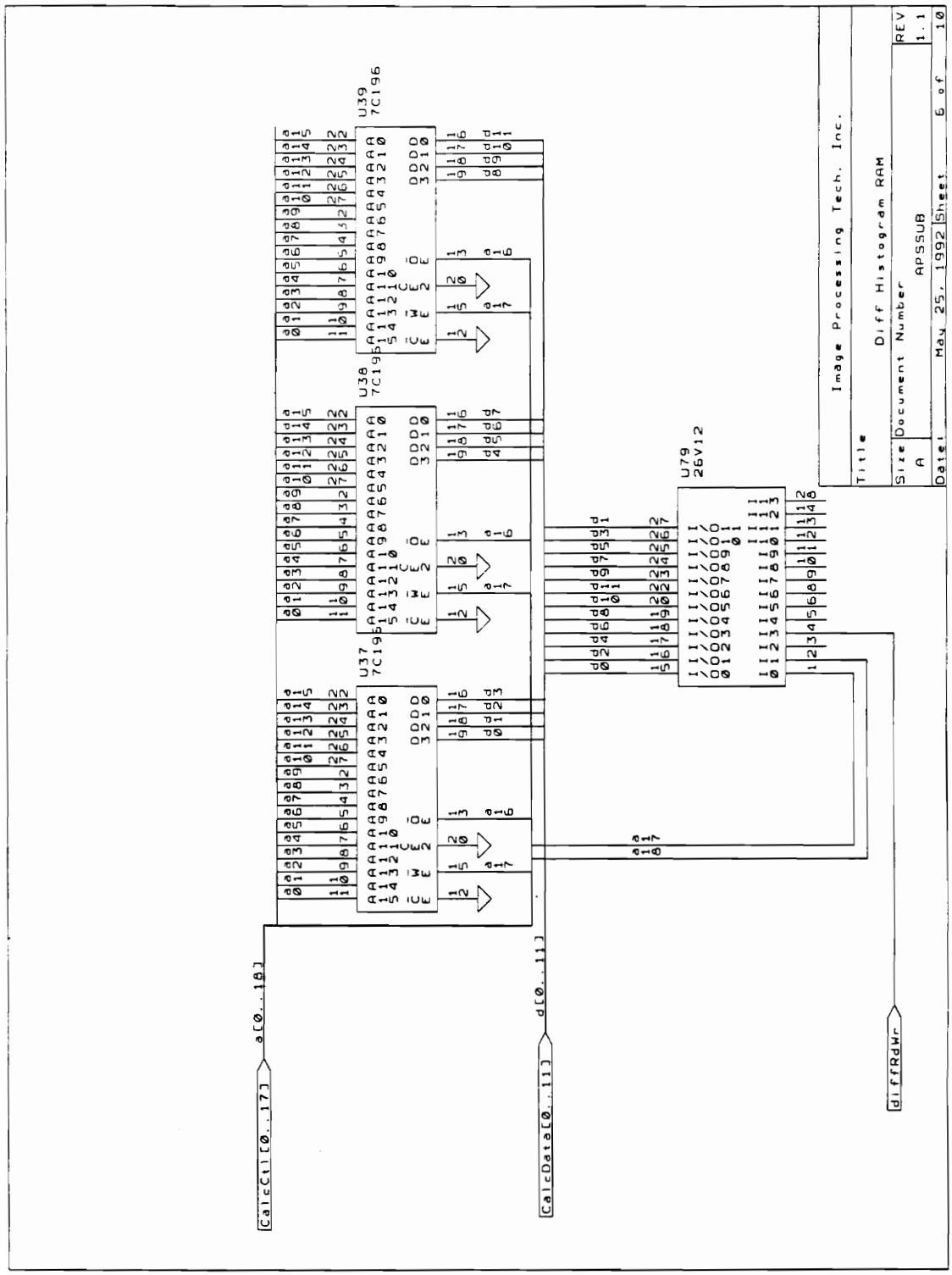
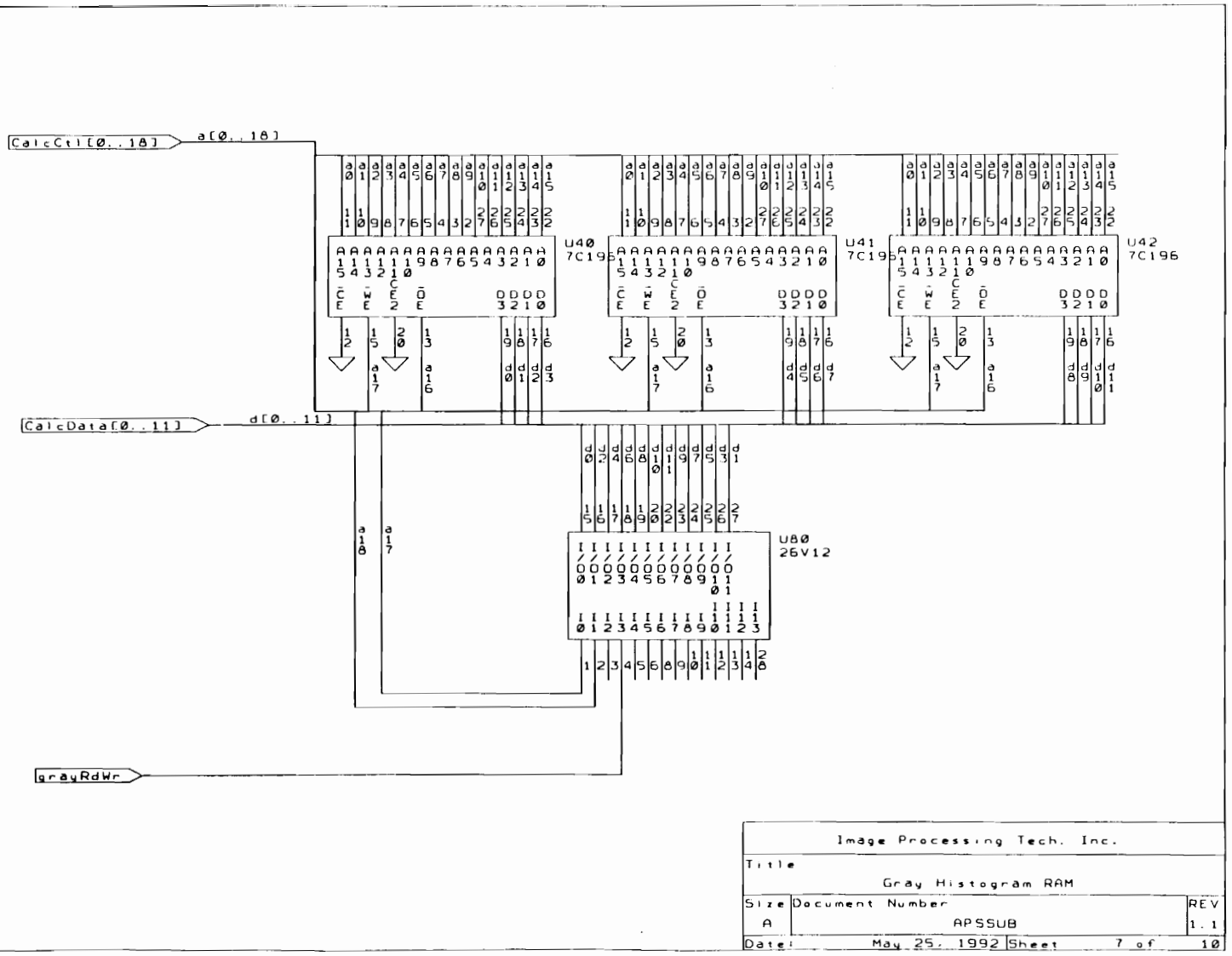
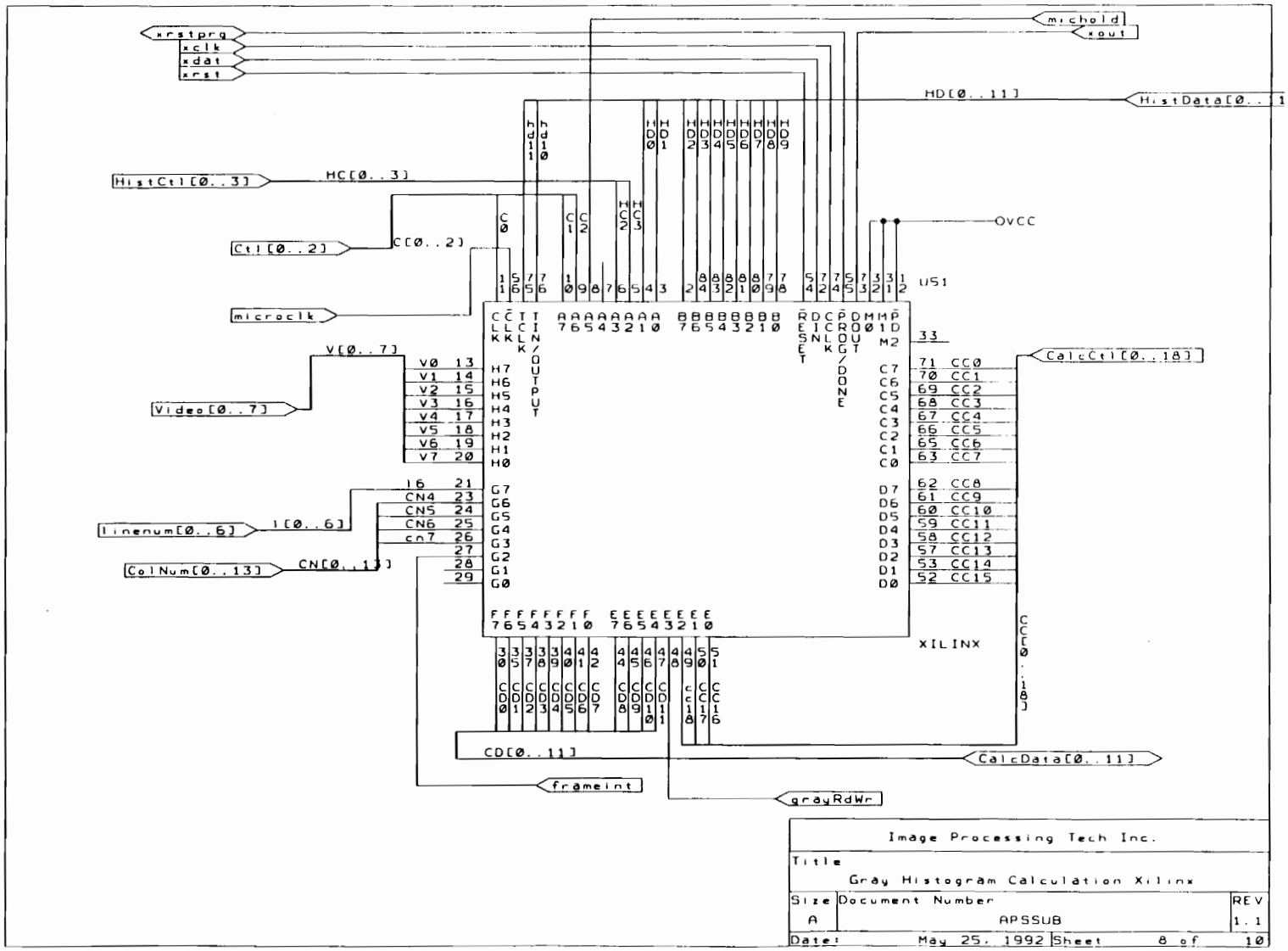
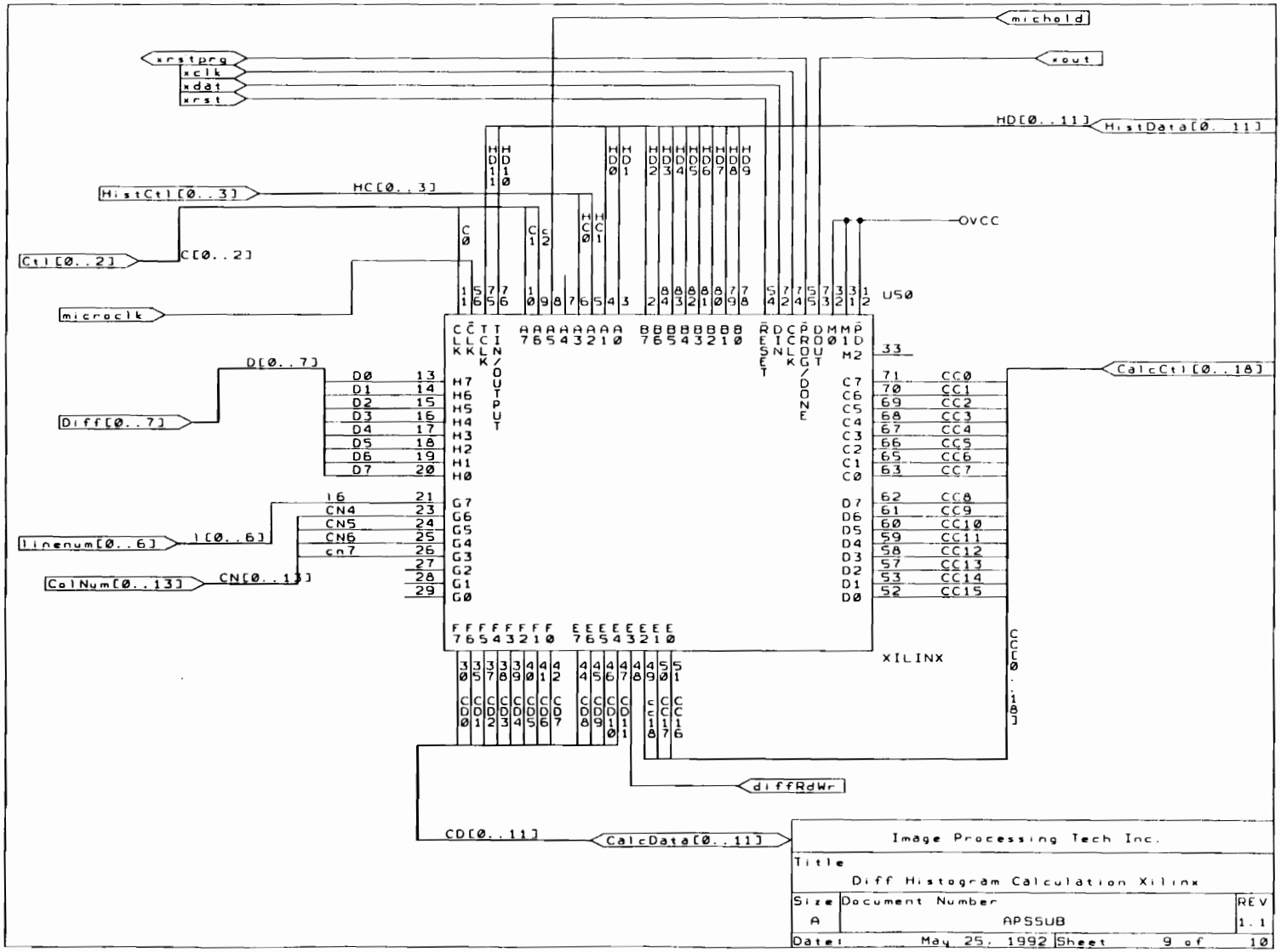
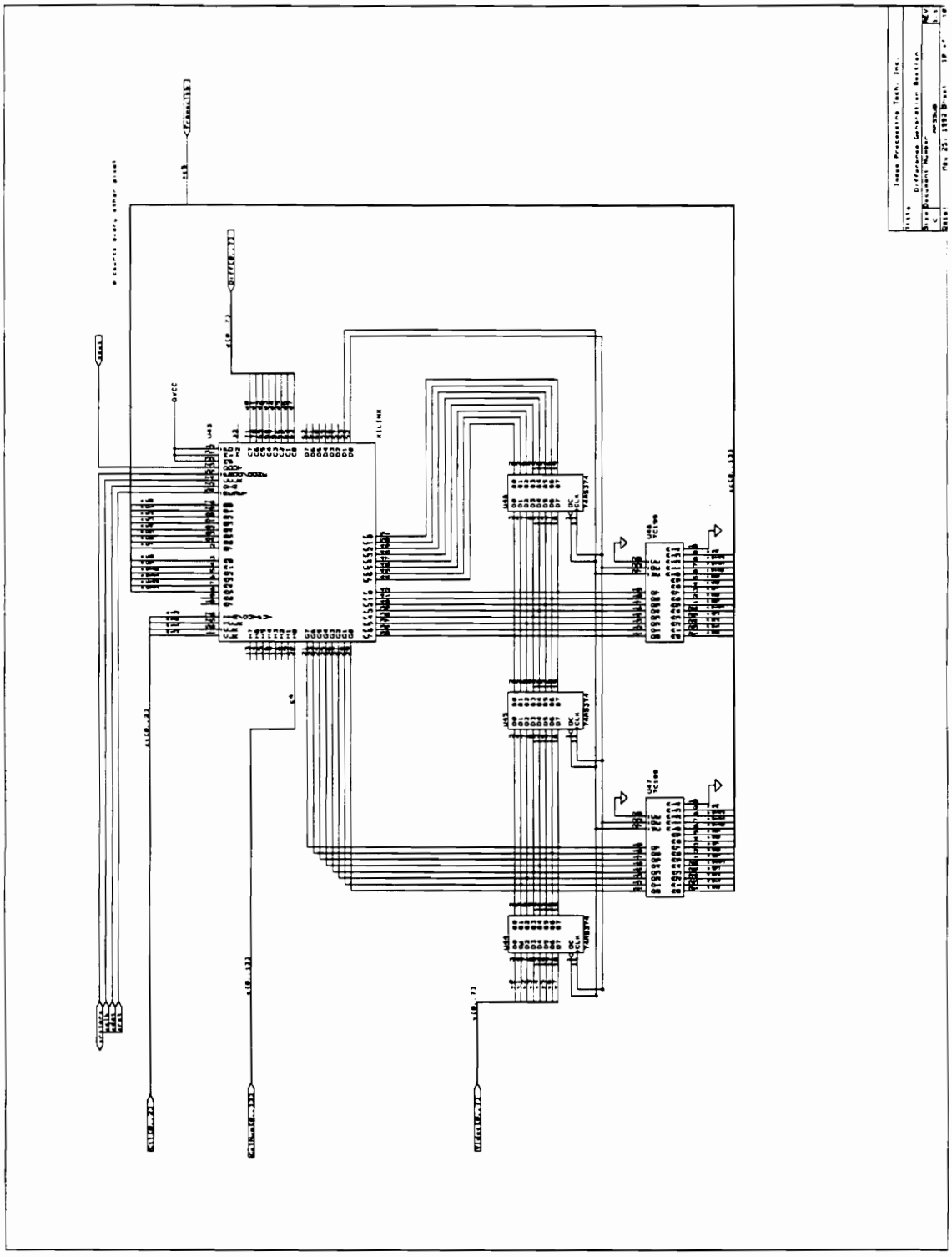


Image Processing Tech. Inc.
Title
Diff Histogram RAM
Size
A
Document Number
APSSUB
REV
1.1
Date: May 25, 1992 Sheet 6 of 10
U37
7C196
U38
7C196
U39
7C196
U79
26V12
CalcCtl[0..17]
CalcData[0..11]
a[0..18]
d[0..11]
diffRdWr


CalcCtl[0..18]
a[0..18]
U40
7C196
U41
7C196
U42
7C196
CalcData[0..11]
d[0..11]
U80
26V12
grayRdWr
Image Processing Tech. Inc.
Title
Gray Histogram RAM
Size
A
Document Number
APSSUB
REV
1.1
Date: May 25, 1992
Sheet 7 of 10


xrstprg
xclk
xdat
xrst
michold
xout
HD[0..11]
HistData[0..11]
HistCtl[0..3]
HC[0..3]
Ctl[0..2]
C[0..2]
OVCC
microclk
US1
V[0..7]
Video[0..7]
linenum[0..6]
L[0..6]
ColNum[0..13]
CN[0..13]
CalcCtl[0..18]
CalcData[0..11]
CD[0..11]
frameint
grayRdWr
XILINX
Image Processing Tech Inc.
Title
Gray Histogram Calculation Xilinx
Size
A
Document Number
APSSUB
REV
1.1
Date: May 25, 1992
Sheet 8 of 10


michold
xrstprg
xclk
xdat
xrst
xout
HD[0..11]
HistData[0..11]
HistCtl[0..3]
HC[0..3]
OVCC
Ctl[0..2]
C[0..2]
microclk
U50
D[0..7]
Diff[0..7]
CalcCtl[0..18]
linenum[0..6]
L[0..6]
ColNum[0..13]
CN[0..13]
XILINX
diffRdWr
CD[0..11]
CalcData[0..11]
Image Processing Tech Inc.
Title
Diff Histogram Calculation Xilinx
Size
A
Document Number
APSSUB
REV
1.1
Date:
May 25, 1992
Sheet
9 of
10



12. Appendix E

Following are the standard PALASM format PAL descriptions.

```
TITLE VIDEO DELAY CONTROLS 2
pattern apsu12
revision 1.0
author Nikos Asimopoulos - Brian Cruikshank
company Image Processing Tech. Inc.
date 2-14-91

chip u12 pal22v10
inclk clk  lv   fv2  mclk  cl13  cl1  cl0 nc  nc  nc  gnd
nc   dclk  outclk outoed outoec outoeb outoea cas ras roe coe vcc

equations
dclk    = /clk
/outoea := /outoed * lv
/outoeb := lv*outoea*outoeb*outoec*outoed + /outoea*lv
/outoec := /outoeb * lv
/outoed := /outoec *lv
/outclk  = /outoed*/clk
/cas    := /outoec + /outoed
/ras    := /outoeb + /outoec + /outoed
/roe     = /outoeb*/clk + /outoec*clk + /outoeb*dclk
/coe     = /outoec*/clk + /outoed*clk + /outoec*dclk
```

```
TITLE VIDEO DELAY CONTROLS 1
pattern not used
revision 1.0
author  Nikos Asimopoulos - Brian Cruikshank
company Image Processing Tech. Inc.
date 2-14-91

chip u13 pal22v10
inclk clk lv  fv2 mclk cl13  cl1    cl0    nc     nc     nc   gnd
nc    s0  s1  s2  s3   outen inclka inclkb inclkc inclkd lclk vcc

equations
lclk    = /clk
inclka := lv*/inclka*/inclkb*/inclkc*/inclkd + inclkd*lv
inclkb := inclka * lv
inclkc := inclkb * lv
inclkd := inclkc * lv
outen   = /inclkd
;/s0    = cl13* cl0*cl1 *clk1
;/s1    = cl13*( /cl0*cl1 + cl0*/cl1 * /clk)
;/s2    = /cl13* cl0*cl1 *clk
;/s3    = /cl13*( /cl0*cl1 + cl0*/cl1 * /clk)
/s0     = inclkd * /clk
s1      = /inclkc
```

```
TITLE CHIP SELECT 2
pattern not used
revision 1.0
author Brian Cruikshank
company Image Processing Tech. Inc.
date

chip u53 palce16v8
mclk rd     wrbar   pa0 pa1 pa2 pa3  pa4    d0b  gnd
oe   stestin misc oc0 oc1 oc2 xrst oselect xclk vcc

equations
xclk: = pa4*/pa3*/pa2*pa1*/pa0*/wrbar*d0b +
        xclk*/(pa4*/pa3*/pa2*pa1*/pa0*/wrbar)
xrst: = pa4*/pa3*/pa2*pa1*pa0*/wrbar*d0b +
        xrst*/(pa4*/pa3*/pa2*pa1*pa0*/wrbar)
oselect: = pa4*pa3*pa2*/pa1*/pa0*/wrbar*d0b +
        oselect*/(pa4*pa3*pa2*/pa1*/pa0*/wrbar)
/misc = /pa4*pa3*/pa2*pa1*pa0*rd
/stestin = /pa4*/pa3*/pa2*pa1*pa0*rd
oc0: = pa4*pa3*pa2*pa1*pa0*/wrbar*d0b + oc0*/(pa4*pa3*pa2*pa1*pa0*/wrbar)
oc1: = pa4*pa3*pa2*pa1*/pa0*/wrbar*d0b + oc1*/(pa4*pa3*pa2*pa1*/pa0*/wrbar)
oc2: = pa4*pa3*pa2*/pa1*pa0*/wrbar*d0b + oc2*/(pa4*pa3*pa2*/pa1*pa0*/wrbar)
```

```
TITLE CHIP SELECT 3
pattern not used
revision 1.0
author Brian Cruikshank
company Image Processing Tech. Inc.
date

chip u55 palce16v8
mclk d0b  wrbar pa0 pa1 pa2 pa3 pa4 cs  gnd
d0  ts1 ts2  hc0 hc1 hc2 hc3 ram xdat vcc

string dstate  ' /hc1 '
string dreset  ' pa4*/pa3*pa2*pa1*pa0 '
string dset    ' pa4*pa3*/pa2*/pa1*pa0 '
string dspace  ' /pa4*pa3*/pa2*/pa1*pa0 '

string gstate  ' /hc3 '
string greset  ' pa4*/pa3*pa2*pa1*/pa0 '
string gset    ' pa4*pa3*/pa2*/pa1*/pa0 '
string gspace  ' /pa4*pa3*/pa2*/pa1*/pa0 '

equations
/ram = pa4*/pa3*/pa2*/pa1*/pa0
xdat: = pa4*/pa3*/pa2*/pa1*pa0*/wrbar*d0b
 + xdat*/(pa4*/pa3*/pa2*/pa1*pa0*/wrbar)
; diff
/hc1: = dstate*/(dreset*/wrbar) + dset*/wrbar
/hc0 = dspace*wrbar
; gray
/hc3: = gstate*/(greset*/wrbar) + gset*/wrbar
/hc2 = gspace*wrbar
; thresh
/ts1 = /pa4*pa3*/pa2*pa1*/pa0*/wrbar
```

```
TITLE WAIT STATE CONTROL AND CHIP SELECT 1
pattern not used
revision 1.0n
author Brian Cruikshank
company Image Processing Tech. Inc.
date

chip u56 pal22v10
mclk rd wrbar pa0 pa1 pa2 pa3 pa4 michold michold2 prgdone gnd
nc hold c2 c1 c0 wr2 clkhldff2 clkhldff fasten serial rom vcc global

string serialad '/pa4*/pa3*/pa2*/pa1*pa0'
string romad    '/pa4*/pa3*/pa2*/pa1*/pa0'
string stestad  '/pa4*/pa3*/pa2*pa1'
string histad   '/pa4*pa3*/pa2*/pa1'
string threshad '/pa4*pa3*/pa2*pa1*/pa0'
string miscad   '/pa4*pa3*/pa2*pa1*pa0'

string startwait '/pa4*/c0*/c1*/c2*/clkhldff*/clkhldff2'
string waiting '(clkhldff + clkhldff2)'

equations
clkhldff2: = clkhldff
clkhldff: = startwait + c0 + c1 + c2
          + /michold + /michold2
c2: = (threshad + miscad + serialad + stestad)*startwait +
    c2*(c1 + c0)
c1: = (serialad)*startwait +
    c2*/c1*/c0 + c1*c0
c0: = (histad)*startwait +
    (c2 + c1)*/c0
/fasten = waiting
hold = startwait*/(threshad*/clkhldff) + clkhldff
/rom = (waiting)*romad*rd
/wr2 = clkhldff*/wrbar
/serial = (waiting)*serialad*(rd + /wrbar)
```

```
TITLE RISC CLOCK GENERATOR
pattern not used
revision 1.0
author Brian Cruikshank
company Image Processing Tech. Inc.
date

chip u59 palce16v8
osc  oscb holdb reset   nc    nc    nc nc nc gnd
hold clk2 clk1  mclk clk25 hold2 oscc resetb analyzer vcc

equations
resetb: = reset + resetb
oscc = oscb
clk25: = /clk25
clk1 = clk25 + holdb + oscb*/clk25
/clk2 = clk25*oscc*/hold2*/holdb
hold2: = holdb
mclk = clk25
analyzer = clk25 + holdb
```

```
TITLE 32 BIT ACCESS CONTROL
pattern not used
revision 1.0
author Brian Cruikshank / Jim Tatem
company Image Processing Tech. Inc.
date 3/4/91

chip u66 palce16v8
a0 a1 wrbar rd rom ram   size0 size1 a0b gnd
nc s0 s1    s2 s3  ramoe ramwe nc    nc vcc

equations
/s0 = /size0*/size1*a0b*a1  +  size0*/size1*/a0b*a1  +  /size0*size1*/a0b*/a1  +
   size0*size1*/a0b*/a1
/s1 = /size0*/size1*/a0b*a1  +  size0*/size1*/a0b*a1  +  /size0*size1*/a0b*/a1  +
   size0*size1*/a0b*/a1
/s2 = /size0*/size1*a0b*/a1  +  size0*/size1*/a0b*/a1  +  /size0*size1*/a0b*/a1  +
   size0*size1*/a0b*/a1
/s3 = /size0*/size1*/a0b*/a1  +  size0*/size1*/a0b*/a1  +  /size0*size1*/a0b*/a1  +
   size0*size1*/a0b*/a1
/ramoe = rd*/ram
/ramwe = /wrbar*/ram
```

```
TITLE HISTOGRAM INCREMENTER
pattern not used
revision 1.0
author  Nikos Asimopoulos
company Image Processing Tech. Inc.
date 3-14-91

chip u79 palce26v12
oe   sel  nc  clk1 nc  nc  vcc nc  nc  nc  nc  nc  nc  nc
d0   d2   d4  d6   d8  d10 gnd d11 d9  d7  d5  d3  d1  nc
glb  o0   o2  o4   o6  o8  o10 o11 o9  o7  o5  o3  o1

equations

d0  := (/d0) * /sel

d1  := ((d1  * /d0) +
        (/d1 * d0)) * /sel

d2  := ((d2  * /(d1*d0)) +
        (/d2 * (d1*d0))) * /sel

d3  := ((d3  * /(d2*d1*d0)) +
        (/d3 * (d2*d1*d0))) * /sel

d4  := ((d4  * /(d3*d2*d1*d0)) +
        (/d4 * (d3*d2*d1*d0))) * /sel


d5  := ((d5  * /(d4*d3*d2*d1*d0)) +
        (/d5 * (d4*d3*d2*d1*d0))) * /sel

d6  := ((d6  * /(d5*d4*d3*d2*d1*d0)) +
        (/d6 * (d5*d4*d3*d2*d1*d0))) * /sel

d7  := ((d7  * /(d6*d5*d4*d3*d2*d1*d0)) +
        (/d7 * (d6*d5*d4*d3*d2*d1*d0))) * /sel

d8  := ((d8  * /(d7*d6*d5*d4*d3*d2*d1*d0)) +
        (/d8 * (d7*d6*d5*d4*d3*d2*d1*d0))) * /sel

d9  := ((d9  * /(d8*d7*d6*d5*d4*d3*d2*d1*d0)) +
        (/d9 * (d8*d7*d6*d5*d4*d3*d2*d1*d0))) * /sel

d10 := ((d10 * /(d9*d8*d7*d6*d5*d4*d3*d2*d1*d0)) +
        (/d10* (d9*d8*d7*d6*d5*d4*d3*d2*d1*d0))) * /sel


d11 := ((d11 * /(d10*d9*d8*d7*d6*d5*d4*d3*d2*d1*d0)) +
        (/d11* (d10*d9*d8*d7*d6*d5*d4*d3*d2*d1*d0))) * /sel
```

```
d0.TRST = /oe
d1.TRST = /oe
d2.TRST = /oe
d3.TRST = /oe
d4.TRST = /oe
d5.TRST = /oe
d6.TRST = /oe
d7.TRST = /oe
d8.TRST = /oe
d9.TRST = /oe
d10.TRST = /oe
d11.TRST = /oe


d0.CLKF = clk1
d1.CLKF = clk1
d2.CLKF = clk1
d3.CLKF = clk1
d4.CLKF = clk1
d5.CLKF = clk1
d6.CLKF = clk1
d7.CLKF = clk1
d8.CLKF = clk1
d9.CLKF = clk1
d10.CLKF = clk1
d11.CLKF = clk1
```

```
TITLE SCANNER CLOCKS
pattern not used
revision 1.0
author  Nikos Asimopoulos - Brian Cruikshank
company Image Processing Tech. Inc.
date 4-22-91

chip u82 palce16v8
fv lv  clk  ofv  olv    oclk    oselect fv2 nc gnd
nc fvo lvo  clko clko1  clko2   lv_n      nc  nc vcc

equations
fvo   = oselect*/ofv  +  /oselect*/fv2
/lv_n = oselect*/olv  +  /oselect*/lv
lvo   = oselect*/olv  +  /oselect*/lv
clko1 = oselect*oclk  +  /oselect*clk
clko  = clko1
```

Vita

Brian Cruikshank was born in Hendersonville, North Carolina. He remained there for most of his childhood with the exception of a few years. At sixteen, he received the Eagle Scout award from the Boy Scouts of America. Afterwards, two years were spent away from North Carolina in Middletown, New Jersey where he graduated from high school. Since then, he has spent the rest of his life working at Image Processing Technologies and pursuing college degrees. In 1988, he received his Bachelor of Science in Electrical Engineering with Honors, and hopefully in 1992, he will receive his Master of Science in Electrical Engineering.